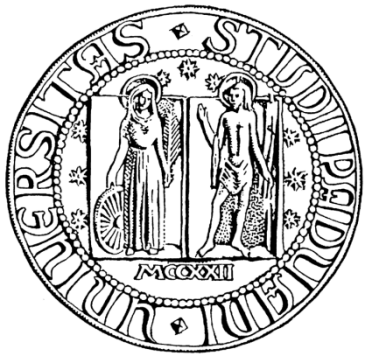

UNIVERSITÀ DEGLI STUDI DI PADOVA

Facoltà di Ingegneria

Corso di laurea in Telecomunicazioni

TESINA DI LAUREA

Studio sui Factor Graph

RELATORE

Prof. ANDREA ZANELLA

Laureando

ANDREA MASSIMILIANO TUZIA

ANNO ACCADEMICO 2010-2011

Indice

Introduzione

I modelli grafici hanno sempre avuto un ruolo importante nello studio ingegneristico poiché offrono una metodologia generale per affrontare i problemi delle applicazioni statistiche soprattutto nel campo delle telecomunicazioni. Negli anni recenti è diventato evidente come statistica e scienza della computazione stiano strettamente collegando i loro obiettivi: da una parte gli studiosi statistici stanno aumentando l'interesse per aspetti computazionali teorici e pratici di modelli e procedure di inferenza; dall'altra la scienza della computazione si sta concentrando su sistemi che interagiscono con il mondo esterno e interpretano i dati sconosciuti in termini di modelli probabilistici. L'area in cui questa tendenza è più evidente è proprio quella dei modelli grafici probabilistici.

I due modelli grafici più famosi sono i Bayesian Netwok (BN) e i Markov Random Fields (MRF) che hanno come obiettivo principale quello di esprimere l'indipendenza condizionata e la fattorizzazione di distribuzioni di probabilità congiunte e di formulare algoritmi di inferenza statistica.

In questo contesto un modello di enorme interesse nel campo delle reti di telecomunicazioni è quello dei Factor Graph (FG), si tratta di nuove strutture matematiche che riuniscono in se le proprietà dei BN e MRF, modellando però graficamente anche quelle strutture dove questi ultimi risultano inadeguati.

Lo scopo di questa tesi è di introdurre il modello dei grafi fattoriali e di descrivere un algoritmo di message-passing, il Sum-Product Algorithm (SPA), che opera sui FG per calcolare funzioni marginali e da cui derivano un enorme quantità di algoritmi pratici.

I FG hanno origine nella teoria dei codici, essendo una generalizzazione dei "Tanner Graph" di Wiberg, che furono introdotti per descrivere famiglie di codici. I FG però fanno un passo in avanti rispetto ai loro predecessori, ovvero applicano questo modello teorico-grafico alle funzioni.

I FG sono grafi bipartiti che offrono una descrizione grafica naturale della fattorizzazione di una funzione globale complessa di molte variabili nel prodotto di funzioni locali più semplici, esprimendo quali variabili siano argomenti di quali funzioni locali.

Nella prima parte della tesina è data la definizione generale e la descrizione della struttura dei FG e le trasformazioni per realizzare delle forme equivalenti di grafi ciclici, utili per modificare strutture di grafi che non permettono il calcolo esatto di funzioni marginali. I FG infatti rappresentano un approccio semplice da descrivere che dà però un consistente vantaggio pedagogico: essi possono essere usati in una vasta serie di campi di

applicazione, offrendo una notazione attrattiva per una grande varietà di problemi di elaborazione, comunicazione e intelligenza artificiale; inoltre attraverso il loro impiego si potranno comprendere un grande numero di algoritmi apparentemente differenti, sviluppati nella scienza computazionale e nell'ingegneria.

Tuttavia i modelli grafici sono spesso associati a un algoritmo particolare e specifico. Nel nostro caso quello di più efficacia e che sfrutta meglio la fattorizzazione rappresentata dai FG è SPA. Questo algoritmo è introdotto nella seconda parte della tesina e consente, attraverso una semplice regola, di calcolare varie funzioni marginali derivate dalla funzione globale. In questo senso la struttura grafica dei FG dà un grande vantaggio nel risolvere problemi di inferenza, cioè, specificando le relazioni di indipendenza condizionata tra i nodi, si arriva a calcolare la distribuzione di probabilità a posteriori delle variabili "nascoste" , deducendola dall'osservazione del modello grafico e dalle variabili osservate.

Operando in appropriati FG, molti algoritmi sviluppati nei processi di segnali e comunicazioni digitali, si possono derivare come istanza specifica di SPA.

Infine i FG forniscono una grande flessibilità nel modellare sistemi e nell'ultima parte della tesina ne vengono presentati degli esempi applicativi sia per quello che riguarda sistemi probabilistici che "di comportamento".

I FG quindi rappresentano un efficace strumento di analisi in un ampio spettro di aree applicative e offrono una visualizzazione grafica con il potenziale di unificare l'elaborazione e il modello di segnali che sono spesso trattati separatamente nei sistemi attuali.

Capitolo 1

Factor Graph

In questo capitolo verranno introdotti i FG nella loro forma originale. Esistono infatti diverse rappresentazioni grafiche alternative, come i Forney-style FG, conosciuti anche come FG normali, che si allontanano dalla notazione generale per alcuni particolari.
E' bene anche introdurre anticipatamente le funzioni che andremo a trattare: considereremo funzioni multivariabili reali $g(x_1, \dots, x_n)$, che chiameremo "funzioni globali", con dominio (o alfabeto) $S = A_1 \times A_2 \times \dots \times A_n$ e codominio R. S viene chiamato *spazio di configurazione* e ogni elemento di S è una configurazione particolare delle variabili: l'elemento $s = (s_1, s_2, \dots, s_n) \in S$ con $s_i \in A_i$ è equivalente all'assegnazione $x_1 = s_1, x_2 = s_2, \dots, x_n = s_n$ e viceversa.

1.1 Definizione

Supponiamo di avere una certa funzione $g(x_1, \dots, x_n)$ che si fattorizza nel prodotto di diverse funzioni locali, aventi come argomento sottoinsiemi di $\{x_1, \dots, x_n\}$:

$$g(x_1, \dots, x_n) = \prod_{j \in J} f_j\left(X_j\right) \quad (1.1)$$

dove J è un insieme discreto e X_j rappresenta un sottoinsieme proprio di $\{x_1, \dots, x_n\}$.
Un FG è un grafo bipartito che esprime la fattorizzazione (1.1). Un FG ha un *nodo variabile* per ogni x_i, un *nodo fattore* per ogni funzione locale f_j e un *arco* di connessione tra i due tipi di nodi se e solo se x_i è argomento di f_j.
Supponiamo di avere una funzione di cinque variabili $g(x_1, x_2, x_3, x_4, x_5)$ che può essere espressa come prodotto di cinque fattori (funzioni locali):

$$g(x_1, x_2, x_3, x_4, x_5) = f_A(x_1, x_2, x_3) f_B(x_1, x_4) f_C(x_2) f_D(x_3) f_E(x_4, x_5) \quad (1.2)$$

per cui $J = \{A, B, C, D, E\}$ e $X_A = \{x_1, x_2, x_3\}, X_B = \{x_1, x_4\}, X_C = \{x_2\}, X_D = \{x_3\}, X_E = \{x_4, x_5\}$. Il corrispondente FG è mostrato in Figura 1.1.

1.2 Struttura

La struttura dei FG eredità tutte le proprietà e le caratteristiche come orientamento, ciclicità e connessione. Tali proprietà danno la possibilità di costruire diverse rappresentazioni della stessa struttura grafica e rendono i FG uno strumento del tutto generale e flessibile, equivalente a diversi modelli grafici alternativi. Ad esempio una BN può essere trasformata nel corrispondente FG orientato o non orientato e lo stesso vale per un MRF, senza perdere alcuna informazione sul modello considerato, ma anzi, rendendo la fattorizzazione più specifica. Tuttavia, essendo interessati al calcolo di funzioni marginali, il modello grafico che legheremo ai FG è quello delle "espressioni ad albero" (*expression tree*), poiché nella scienza della computazione la funzione marginale viene spesso rappresentata come un albero ordinato con radice.

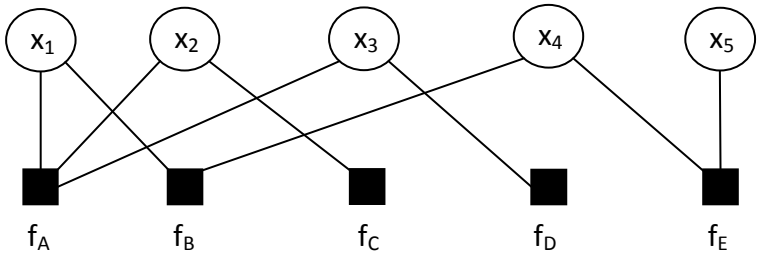

Figura 1.1. *Factor graph per la funzione (1.2).*

1.2.1 Expression tree

In molte situazioni siamo interessati a calcolare da una funzione di probabilità congiunta, una particolare funzione marginale. La sua forma usando (1.2), sarà del tipo:

$$g_1(x_1) = \left(\sum_{x_2 \in S_2} f_C(x_2) \left(\sum_{x_3 \in S_3} f_A(x_1, x_2, x_3) f_D(x_3) \right) \right) \cdot \left(\sum_{x_4 \in S_4} f_B(x_1, x_4) \left(\sum_{x_5 \in S_5} f_E(x_4, x_5) \right) \right). \quad (1.3)$$

La rappresentazione di espressioni aritmetiche come questa viene fatta attraverso alberi ordinati dotati di radice che vengono chiamati *expression tree*.

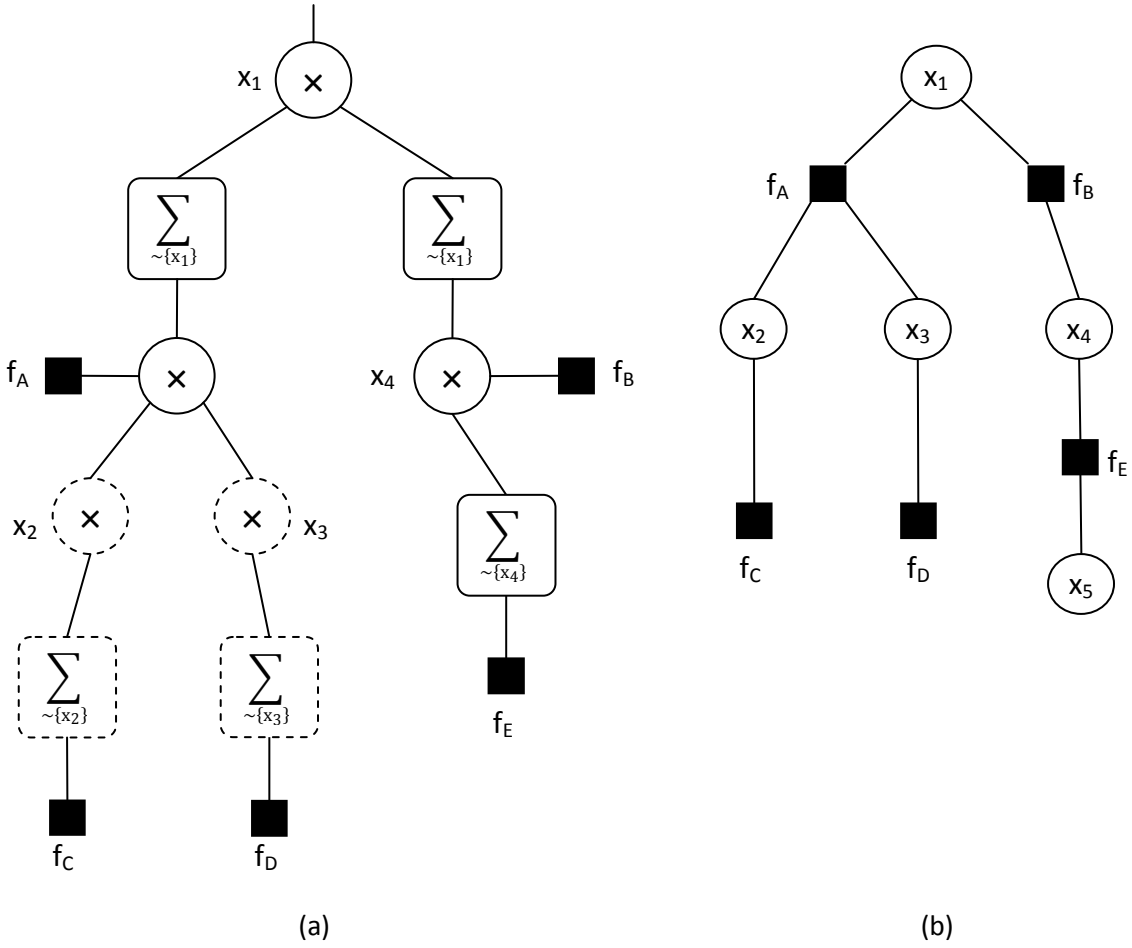

Figura 1.2. *(a) Rappresentazione di (1.3) attraverso una espressione albero. (b) FG di Figura 1 ridisegnato come albero con radice.*

In queste strutture i vertici interni rappresentano gli operatori matematici, mentre i nodi esterni le variabili. Estendendo questa forma a funzioni, cioè rappresentando le funzioni locali di (1.2) nei vertici esterni di una espressione ad albero, è facile notare la corrispondenza tra gli expression tree e i FG mettendo a confronto i due modelli. Esistono infatti semplici regole di conversione per passare da un modello all'altro.
In Figura 1.2(b) è rappresentato il FG di Figura 1.1 come un expression tree con radice x_1. Sostituiamo ogni nodo variabile del FG con un operatore prodotto e ogni nodo fattore con una coppia operatore prodotto moltiplicato da f. Infine tra un nodo fattore e il suo unico padre x_i inseriamo un operatore $\sum_{\sim\{x_i\}}$, che rappresenta una notazione non standard e indica una sommatoria nidificata rispetto a tutte le variabili ad eccezione di x_i. Tutte queste trasformazioni locali sono illustrate in Figura 1.2. Come si può notare ci sono degli operatori nell'espressioni ad albero che possono essere eliminati: prodotti trascurabili nei nodi esterni (con uno o nessun operando) possono essere omessi, così come operatori $\sum_{\sim\{x_i\}}$ applicati a funzioni con singolo argomento x_i.

1.2.2 FG aciclici e calcolo di una singola funzione marginale

Abbiamo visto precedentemente lo stretto legame che sussiste tra un FG e le corrispondenti expression tree, che rappresentano un *algoritmo* per calcolare un espressione matematica. La forma più importante di FG sono i FG aciclici. Per questi tipi di FG si può fare una osservazione di fondamentale rilevanza: *quando un facor graph è privo di cicli, esso codifica nella sua struttura non solo la fattorizzazione della funzione globale, ma anche espressioni aritmetiche con le quali le funzioni marginali associate alla funzione globale possono essere calcolate*.
Prima di vedere l'algoritmo generale associato ai FG, ovvero il SPA, descriveremo ora un suo caso particolare che calcola la funzione marginale $g_i(x_i)$ in un FG aciclico con x_i come radice.
L'algoritmo si sviluppa con una procedura "bottom-up", cioè che parte dai nodi foglia del FG per poi risalire di figlio in padre fino alla radice. E' importante caratterizzare le funzioni dei vari nodi, distinguendo i nodi esterni da quelli interni. La procedura inizia dai nodi esterni: i nodi variabile esterni x_j mandano un messaggio "funzione identità" al loro padre, mentre i nodi fattore esterni f_h mandano una descrizione della funzione f_h associata. I nodi interni invece prima di calcolare e passare il messaggio al proprio padre, devono aspettare di ricevere i messaggi da tutti i figli: un nodo variabile interno "inoltrerà"

semplicemente il prodotto dei messaggi ricevuti dai propri figli; un nodo fattore interno f_h con figli $x_j \in X_h$ calcola la sommatoria $\sum_{x_j \in X_h}$ del prodotto di f_h coi messaggi ricevuti. Il calcolo termina alla radice del FG, dove è ottenuta la funzione marginale $g_i(x_i)$ come prodotto dei messaggi ricevuti a x_i.
E' importante notare che i messaggi passati su un arco $\{x, f\}$, sia dalla variabile x al nodo fattore f sia viceversa, da f a x, sono funzioni o meglio, prodotti di funzioni, con un singolo argomento x, che è la variabile associata all'arco.

1.3 Trasformazioni di FG ciclici

La procedura di calcolo descritta precedentemente termina solo se il FG è un albero, i.e. se non contiene cicli. In molte applicazioni tuttavia si è costretti a trattare con FG cilici, come ad esempio nella codifica di turbo code o Low Density Parity Check (LDPC) code, sui quali un algoritmo iterativo non ha una terminazione naturale e il procedimento di message-passing porta a una propagazione senza fine dei messaggi attorno ai cicli, con lo stesso messaggio passato più volte su uno stesso arco.
Sebbene in casi particolari i messaggi possano convergere, diversamente da quanto visto per i FG aciclici, il valore al quale l'algoritmo converge non può essere interpretato come una funzione marginale esatta, ma solo come una sua approssimazione.
In questa sezione daremo una serie di semplici regole di trasformazione che possono essere applicate a FG con una struttura che non permette il calcolo esatto delle funzioni marginali associate alla funzione globale e vedremo la loro utilità in molti casi. E' sempre possibile infatti trasformare un FG ciclico in una forma priva di cicli, incrementando però la complessità delle funzioni locali e/o dei domini delle variabili.

1.3.1 Clustering

E' sempre possibile raggruppare nodi dello stesso tipo, siano essi nodi fattore che nodi variabile, senza cambiare la fattorizzazione della funzione globale: questa operazione prende il nome di *clustering*, ovvero "raggruppamento". Considereremo qui solo il caso del raggruppamento di due nodi, ma è semplice la generalizzazione a un numero più elevato.
Se v e w sono due nodi che si vogliono raggruppare, semplicemente si eliminano dal FG corrispondente assieme a tutti i loro archi incidenti e si crea al loro posto nel FG un nuovo

nodo che rappresenta la coppia (v, w) e che si connette ai nodi vicini a v e w del FG originario.

Se v e w rappresentano nodi variabile con alfabeto A_v e A_w, la nuova coppia (v, w) sarà un nuovo nodo variabile e avrà dominio dato dal prodotto dei domini originari, $A_v \times A_w$. In questo modo aumenta notevolmente la grandezza dell'alfabeto incrementando la complessità computazionale dell'algoritmo che opera sul grafo. Inoltre ogni funzione locale del FG che aveva come argomento v o w, dovrà essere sostituita da una funzione equivalente f' avente come argomento la coppia (v, w). Questa conversione non ha conseguenze sulla complessità della funzione locale e soprattutto non altera la complessità di calcolo dell'algoritmo.

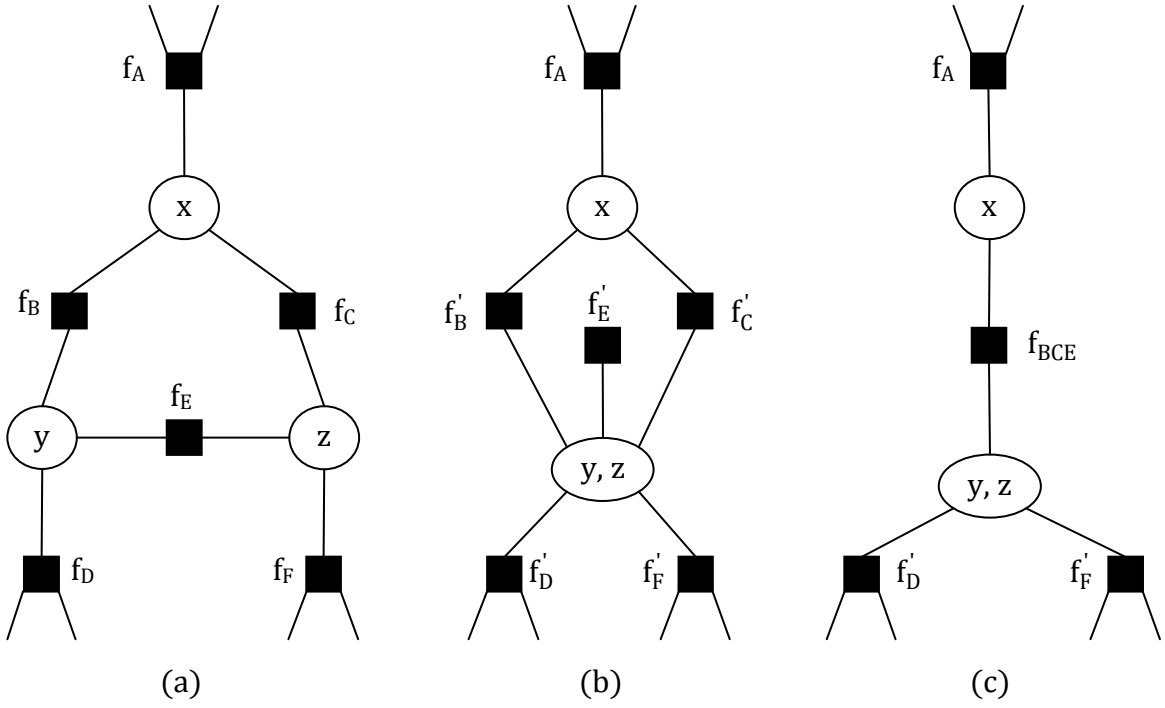

Figura 1.3. *Trasformazione di clustering. (a) FG originario. (b) Raggruppamento dei nodi variabile y e z. (c) Raggruppamento dei nodi fattore f_B, f_C e f_E.*

Per quanto riguarda invece il raggruppamento di nodi fattore con la coppia (v, w), si intende un nuovo nodo fattore che rappresenta il prodotto delle funzioni locali di v e w. Se indichiamo con X_v e X_w gli insiemi degli argomenti delle due funzioni, dalla loro unione $X_v \cup X_w$ si ottiene l'insieme degli argomenti della nuova coppia (v, w). Al contrario quindi del raggruppamento di variabili, il raggruppamento di funzioni aumenterà la complessità delle funzioni locali, lasciando inalterata invece quelle delle variabili.

L'utilità dell'operazione di *clustering* è evidente in quanto può essere usata per eliminare cicli nei FG ciclici come viene raffigurato in Figura 1.3, dove è rappresentato un frammento di FG in cui è presente un ciclo che dopo una serie di trasformazioni di raggruppamento è stato eliminato. Se poi il resto del grafo a cui si fa riferimento è aciclico, si possono calcolare esattamente le funzioni marginali associate alla funzione globale rappresentata dal FG. E' importante infine notare come le funzioni locali dopo operazioni di *clustering* mantengano la dipendenza dal vecchio FG. Infatti una funzione, che aveva originariamente come argomento x_i, dopo un raggruppamento di nodi variabile viene connessa alla coppia (x_i, x_j), ma in realtà non dipende direttamente da x_j. In questo modo la funzione globale rappresentata dal nuovo FG è identica a quella originaria, l'unica differenza è che nella regione in cui il FG è stato trasformato è la complessità dei messaggi dell'algoritmo è maggiore.

1.3.3 Stretching variable node

Nel calcolare una particolare funzione marginale associata a $g(x_1, \dots, x_n)$ abbiamo visto che nel messaggio che attraversa un arco {v, w} i prodotti delle funzioni locali sono sommati per calcolare la marginale della variabile x_i associata all'arco. Al di fuori degli archi incidenti la variabile non esiste alcuna funzione che dipenda da x_i in forma sommatoria, cioè x_i è marginalizzata. Grazie all'operazione di *stretching* si può estendere questa regione, dove la variabile x_i è rappresentata senza essere marginalizzata.
Chiamiamo $n_2(x)$ l'insieme dei nodi che possono essere raggiunti da x seguendo un percorso di lunghezza 2. L'insieme $n_2(x)$ quindi contiene nodi variabile e ogni $y_i \in n_2(x)$ si può sostituire con la coppia (x, y_i) come nel clustering e quindi i nodi fattore incidenti y si modificheranno senza incrementare la loro complessità. Questa trasformazione prende il nome di "allungamento" perché è come se la variabile x si allungasse sulle altre variabili. Inoltre è possibile iterare la trasformazione: se B è l'insieme dei nodi su cui x si è allungata, lo stesso procedimento lo si può effettuare su $n_2(B)$, cioè l'insieme dei nodi variabile raggiungibili da B attraverso un cammino di lunghezza 2.
La proprietà dell'operazione di stretching è che $n_2(x)$ induce un sottografo connesso del FG che genera un ben definito insieme di archi sui quali x è rappresentata senza essere presente in una forma sommatoria. Se l'operazione è usata solo su una variabile, allora le variabili del FG trasformato saranno tutte distinte; se invece si allungano più variabili questo non è più vero.

Alla trasformazione di stretching è collegato in maniera fondamentale il concetto di ridondanza: è possibile infatti che un arco o una variabile dopo una operazione di stretching diventi ridondante. Prendiamo una funzione locale f, un suo arco incidente e e l'insieme delle variabili associate ad e (nel FG originario) X_e Se X_e è contenuto nell'unione degli insiemi delle variabili associate agli archi incidenti a f diversi da e, allora e è ridondante e può essere eliminato dal FG. Se poi tutti gli archi incidenti a una variabile sono ridondanti, allora anche la variabile è ridondante e può venire eliminata assieme ai suoi archi incidenti. E' importante chiarire il fatto che non stiamo eliminando la variabile dal FG, ma solo la sua rappresentazione come nodo variabile che il FG le associa.
E' semplice ora comprendere come dopo sistematiche operazioni di stretching di nodi variabile attorno a cicli, sia possibile eliminare archi e variabili ridondanti, rompendo così il ciclo e trasformando un FG ciclico in una forma equivalente priva di cicli (un semplice esempio viene proposto in Figura 1.4). Tutto ciò può essere effettuato senza incrementare la complessità delle funzioni locali; l'unico costo da pagare è l'aumento della grandezza dell'alfabeto dei nodi variabile che vengono accoppiati e quindi l'incremento della complessità delle variabili.

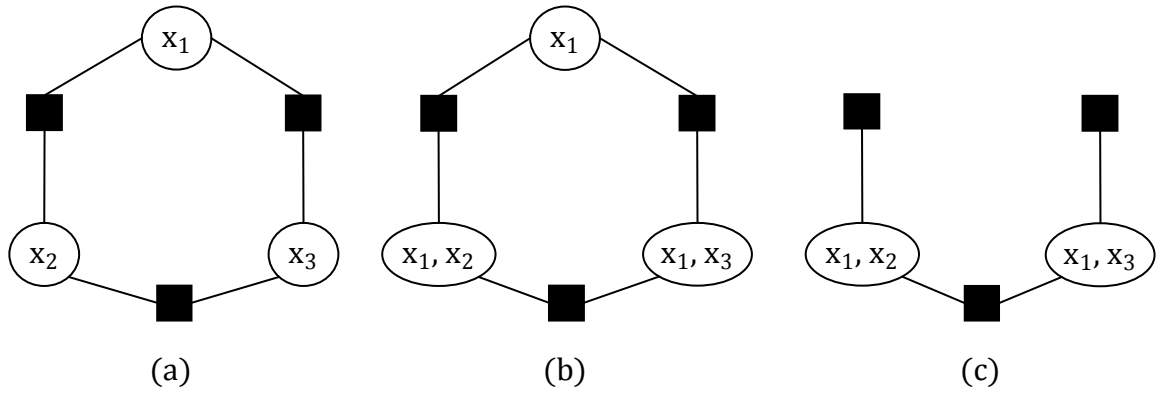

Figura 1.4 *Trasformazione di allungamento. (a) FG originario. (b) Nodo x_1 è allungato sui nodi x_2 e x_3. (c) Rimozione del nodo x_1.*

1.3.4 Spanning tree

Si definisce uno *spanning tree* T di un grafo connsesso G, un sottografo connesso aciclico di G che contiene tutti i nodi del grafo originario. Sia F un FG connesso e T uno spanning tree ricavato da F. Chiamiamo n(x) l'insieme dei nodi fattore che hanno la variabile x per argomento. Dalla teoria dei grafi sappiamo che, poiché T è un albero, esiste un unico cammino tra due nodi di T, in particolare quindi anche per ogni elemento appartenente a n(x). Supponiamo ora di fare una operazione di stretching su x, allungandola su tutti i nodi

variabili che si incontrano per raggiungere i nodi contenuti su n(x) e il FG così trasformato lo chiamiamo F'.

Abbiamo così creato uno spanning tree T e un FG trasformato F'. Dal confronto di questi due modelli grafici si può dimostrare che ogni arco di F' che non è contenuto in T è ridondante e può essere eliminato. Uno spanning tree quindi, essendo per natura aciclico, può essere usato attraverso un algoritmo per calcolare esattamente funzioni marginali associate al FG ciclico da cui è stato ricavato.

Infatti se e è un arco di F' e non di T, chiamiamo X_e l'insieme delle variabili associate ad e ed f la funzione locale su quale e è incidente. Per ogni $x \in X_e$ c'è un cammino in T da f a x, ma x è stata allungata su tutti i nodi variabile lungo questo cammino e in particolare sui vicini di f. Poiché ogni elemento di X_e appare in qualche nodo variabile che non è inciso da e, e è ridondate. La rimozione di e non incide sullo stato di ridondanza degli altri archi in F', in questo modo tutti gli archi ridondanti possono essere eliminati.

La dimostrazione ci è quindi fornita dal fatto che ogni variabile x è allungata sulle variabili che si incontrano nel cammino da x a ogni funzione locale che ha x come argomento. Nella regione di T perciò in cui x è coinvolta non è marginalizzata.

Capitolo 2

The Sum Product Algorithm

Nel calcolo di una particolare funzione marginale associate alla funzione globale, abbiamo implicitamente assunto che i nodi che compongono il FG possano essere considerati come dei processori capaci di trasmettere e ricevere messaggi dagli archi a cui sono connessi e di svolgere delle operazioni con i messaggi ricevuti. In questo capitolo verrà per prima cosa caratterizzato il particolare procedimento di message – passing che il SPA utilizza; poi verrà definito l'algoritmo generale di cui abbiamo già visto una esemplificazione precedentemente nel calcolo di una singola funzione marginale, attraverso il quale si possono calcolare tutte le funzioni marginali associate a una funzione globale.

2.1 The Message – Passing Schedule

Un algoritmo di message passing utilizza la propagazione di messaggi per risolvere attraverso un numero finito di passi un problema. Per quanto riguarda SPA lo scopo è quello di calcolare tutte le funzioni marginali associate a una funzione globale. Nel FG quindi che modella la funzione globale, i nodi devono poter ricevere e trasmettere messaggi attraverso gli archi incidenti al nodo e compiere delle operazioni (prodotti e somme) che coinvolgono i messaggi ricevuti.
Innanzitutto è bene fare delle premesse per inquadrare in modo generale il procedimento: sebbene SPA non richieda che i messaggi inviati dai nodi siano sincronizzati, assumeremo un clock globale a tempo discreto, in cui al massimo un messaggio si può propagare su ogni arco durante un periodo di clock; messaggi successivi inviati sullo stesso arco e nella stessa direzione rimpiazzeranno i messaggi precedenti; inoltre un messaggio da un nodo v al tempo i sarà funzione solo delle funzione locale associata a v (se presente) e dei messaggi precedenti ricevuti da v prima di i. Supporremo anche che per iniziare il procedimento un messaggio contenente la funzione identità sia arrivato attraverso tutti gli archi incidenti a ogni nodo: infatti dato che i messaggi sono funzione dei messaggi precedenti ricevuti dal nodo, solo nei nodi con grado uno è possibile iniziare l'invio di un

messaggio, essendo indipendente da quelli precedenti e se il FG fosse sprovvisto di nodi con grado uno l'algoritmo non partirebbe.
Un *message passing schedule* in un FG è un piano di trasmissione dei messaggi che si propagano sugli archi durante ogni periodo di clock ed esistono diverse *schedule* di trasmissione. Ad esempio la modalità *flooding* permette di trasmette ad ogni periodo di clock un messaggio su ogni arco del FG simultaneamente, avendo così dei vantaggi a livello di velocità di calcolo. Una trasmissione invece in cui al massimo un messaggio si propaga nel grafo ad ogni periodo si chiama *serial schedule*. SPA utilizza una *two-way schedule* dove i messaggi si propagano dai nodi esterni del grafo verso tutti gli altri nodi interni e sono passati una sola volta in entrambe le direzioni di ogni arco del grafo. Questa propagazione è molto conveniente in quanto minimizza il numero totale di messaggi da trasmettere: se si hanno E archi, si avranno ovviamente 2E messaggi da trasmettere.
Si dice nn nodo *v* che ha un messaggio "in attesa" su un arco *e* se ha ricevuto messaggi su tutti gli archi incidenti diversi da *e* dopo l'ultimo arrivo di un messaggio su *e*. La ricezione di un messaggio dall'arco *e* creerà a sua volta messaggi in attesa sugli altri archi incidenti a *v*. Solo i messaggi in attesa hanno bisogno di essere trasmessi, poiché potrebbero essere diversi dai precedenti messaggi sullo stesso arco.
Per una schedule finita, cioè in cui non si raggiunge mai un nodo in cui non ci sono messaggi in attesa e non si innescano mai catene infinte, SPA termina calcolando per ogni x_i il prodotto dei messaggi più recenti ricevuti al nodo variabile x_i.

2.2 L'Algoritmo generale

Abbiamo già visto nel capitolo 1, attraverso il calcolo dello *i*-esimo SPA, le proprietà di base del FG che vengono sfruttate dall'algoritmo generale di message passing: (1) i prodotti delle funzioni locali possono essere raccolti lungo i percorsi nel grafo e (2) le variabili possono essere rappresentate nelle sommatorie fuori la ragione del grafo in cui non sono coinvolte.
SPA calcola tutte le *i*-esime funzioni marginali associate alla funzione globale. Questo calcolo può essere effettuato applicando il SPA separatamente per tutte le variabili x_i ma questo approccio è poco efficiente, in quanto sottocalcoli per diverse funzioni marginali potrebbe essere gli stessi. Il calcolo di tutte le $g_i(x_i)$ può essere effettuato efficientemente ricoprendo invece un singolo FG da tutte le possibili istanze dello *i*-esimo algoritmo.
A differenza del singolo algoritmo, in SPA non viene considerato nessun nodo come radice: non c'è alcuna relazione di parentela padre-figlio o gerarchica tra nodi adiacenti. Lo

sviluppo è "dinamico", ogni nodo vicino *w* a un nodo *v* è considerato a un certo punto come padre di *v* e il calcolo avviene come nell'algoritmo *i*-esimo, cioè trasmettendo il messaggio al padre *w* di *v* e considerando gli altri nodi adiacenti a *v* come figli.

SPA inizia dai nodi esterni del FG, ovvero da quei nodi sui quali è incidente un solo arco. I nodi fattore esterni invieranno come messaggio una rappresentazione della funzione locale associata, mentre i nodi variabile esterni la funzione unità (nella realizzazione pratica non sarebbe necessaria). I nodi interni rimangono inattivi finché non hanno ricevuto i messaggi da tutti i nodi adiacenti, tranne uno. Quando questo accade il nodo *v* è in grado di calcolare il messaggio da inviare sull'unico arco libero e recapitarlo al nodo adiacente *w*, considerato temporaneamente come suo padre. Dopo l'invio il nodo *v* ritorna inattivo, in attesa di un messaggio proveniente da *w*; una volta arrivato il messaggio, il nodo *v* può calcolare nuovi messaggi da recapitare agli altri vicini diversi da *w*, considerati uno alla volta come padre di *v*. Osserviamo che nodi con grado due non eseguono calcoli: un messaggio in entrata sarà semplicemente trasferito come messaggio in uscita. L'algoritmo termina, in accordo con la modalità two way schedule, quando su ogni arco si sono propagati due messaggi, uno per ogni direzioni. Il prodotto dei messaggi entranti in un nodo variabile x_i è la funzione marginale $g_i(x_i)$ cercata.

SPA funziona con la seguente regola di aggiornamento: il messaggio trasmesso da un nodo *v* attraverso un arco *e* è il prodotto di tutti i messaggi ricevuti da *v* sugli altri archi diversi da *e*. Il messaggio trasmesso invece da un nodo fattore f è il prodotto della funzione locale associata a f con tutti i messaggi ricevuti da f sugli altri archi diversi da *e*, sommato rispetto a tutte le variabili eccetto quella associata ad *e*.

Indichiamo con $\mu_{x\to f}(x)$ il messaggio da un nodo variabile x a un nodo fattore f; mentre indichiamo con $\mu_{f\to x}(x)$ il contrario e n(v) l'insieme dei nodi adiacenti a *v*.

Da variabile a funzione locale:

$$\mu_{x\to f}(x) = \prod_{h\in n(x)\setminus\{f\}} \mu_{h\to x}(x). \quad (2.1)$$

Da funzione locale a variabile:

$$\mu_{f\to x}(x) = \sum_{\sim\{x\}} \left(f(X) \prod_{y\in n(f)\setminus\{x\}} \mu_{y\to f}(y) \right) \quad (2.2)$$

dove $X = n(f)$ è l'insieme degli argomenti della funzione f.

L'efficienza di SPA si rileva dal fatto che i prodotti delle funzioni locali raccolti lungo un percorso nel FG possono essere marginalizzati, cioè non è necessario trasportare tutte le variabili lungo un arco. In generale, una variabile ha bisogno di essere trasmessa se è un argomento di una successiva funzione locale.
Infine si osserva che la funzione marginale $g_i(x_i)$ può essere calcolata in maniera equivalente anche come il prodotto dei due messaggi che si propagano su qualsiasi arco incidente al nodo variabile x_i, dato che il messaggio uscente è il prodotto degli altri messaggi entranti nel nodo.

2.3 Un esempio dettagliato

Nella Figura 2.1 è illustrata la procedura del SPA mettendo in evidenza per ogni passo dell'algoritmo la propagazione del flusso di messaggi da ogni nodo. Il procedimento è stato applicato al FG di Figura 1.1, per cui l'algoritmo termina dopo 7 passi.

Passo 1:

$$\mu_{f_C \to x_2}(x_2) = f_C(x_2)$$
$$\mu_{f_D \to x_3}(x_3) = f_D(x_3)$$
$$\mu_{x_5 \to f_E}(x_5) = 1.$$

Passo 2:

$$\mu_{x_2 \to f_A}(x_2) = \mu_{f_C \to x_2}(x_2)$$
$$\mu_{x_3 \to f_A}(x_3) = \mu_{f_D \to x_3}(x_3)$$
$$\mu_{f_E \to x_4}(x_4) = \sum_{\sim\{x_4\}} \mu_{x_5 \to f_E}(x_5) f_E(x_4, x_5).$$

Passo 3:

$$\mu_{f_A \to x_1}(x_1) = \sum_{\sim\{x_1\}} \mu_{x_2 \to f_A}(x_2) \mu_{x_3 \to f_A}(x_3) f_A(x_1, x_2, x_3)$$
$$\mu_{x_4 \to f_B}(x_4) = \mu_{f_E \to x_4}(x_4).$$

Passo 4:

$$\mu_{x_1 \to f_B}(x_1) = \mu_{f_A \to x_1}(x_1)$$
$$\mu_{f_B \to x_1}(x_1) = \sum_{\sim\{x_1\}} \mu_{x_4 \to f_B}(x_4)\, f_B(x_1, x_4).$$

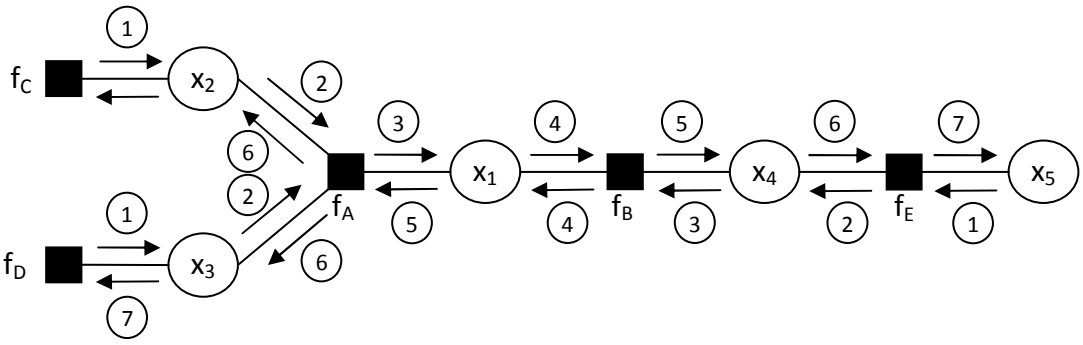

Figura 2.1 *Propagazione dei messaggi generati a ogni passo di Sum-Product algotithm.*

Passo 5:

$$\mu_{f_B \to x_4}(x_4) = \sum_{\sim\{x_4\}} \mu_{x_1 \to f_B}(x_4)\, f_B(x_1, x_4)$$

$$\mu_{x_1 \to f_A}(x_1) = \mu_{f_B \to x_1}(x_1).$$

Passo 6:

$$\mu_{x_4 \to f_E}(x_4) = \mu_{f_B \to x_4}(x_4)$$

$$\mu_{f_A \to x_2}(x_2) = \sum_{\sim\{x_2\}} \mu_{x_1 \to f_A}(x_1)\mu_{x_3 \to f_A}(x_3) f_A(x_1, x_2, x_3)$$

$$\mu_{f_A \to x_3}(x_3) = \sum_{\sim\{x_3\}} \mu_{x_1 \to f_A}(x_1)\mu_{x_2 \to f_A}(x_2) f_A(x_1, x_2, x_3).$$

Passo 7:

$$\mu_{x_2 \to f_C}(x_2) = \mu_{f_A \to x_2}(x_2)$$

$$\mu_{x_3 \to f_D}(x_3) = \mu_{f_A \to x_3}(x_3)$$

$$\mu_{f_E \to x_5}(x_5) = \sum_{\sim\{x_5\}} \mu_{x_4 \to f_E}(x_4) f_E(x_4, x_5).$$

Terminazione:

$$g_1(x_1) = \mu_{f_B \to x_1}(x_1)\mu_{f_A \to x_1}(x_1)$$

$$g_2(x_2) = \mu_{f_c \to x_2}(x_2)\mu_{f_A \to x_2}(x_2)$$

$$g_3(x_3) = \mu_{f_D \to x_3}(x_3)\mu_{f_A \to x_3}(x_3)$$

$$g_4(x_4) = \mu_{f_E \to x_4}(x_4)\mu_{f_B \to x_4}(x_4)$$

$$g_5(x_5) = \mu_{f_E \to x_5}(x_5).$$

Allo stesso modo, invece che calcolare la i-esima funzione marginale come prodotto dei messaggi entranti in un nodo variabile, si può calcolare $g_i(x_i)$ come prodotto dei due messaggi che sono passati su uno stesso arco incedente x_i. Per esempio $g_2(x_2)$ può venire calcolata come:

$$g_2(x_2) = \mu_{f_c \to x_2}(x_2)\mu_{x_2 \to f_C}(x_2) = \mu_{f_A \to x_2}(x_2)\mu_{x_2 \to f_A}(x_2).$$

Capitolo 3

Modellare sistemi attraverso Factor Graph

I FG possono essere utilizzati per modellare sistemi diversi. Nel seguito verranno descritti due possibili approcci: nel primo, il modello di "comportamento" (comportamentale), un FG rappresenta la fattorizzazione della funzione indicatrice di un particolare comportamento del sistema osservato e ne specifica quali configurazioni di variabili siano valide; nel secondo invece, il modello probabilistico, un FG può essere usato per rappresentare la funzione di probabilità di massa congiunta delle variabili che interagiscono nel sistema.
In molte applicazioni è anche possibile utilizzare una combinazione dei due modelli proposti, come ad esempio nella codifica di canale dove le parole di codice valide vengono modellate con il modello di comportamento, mentre l'APP con il modello probabilistico.

3.1 Il modello comportamentale

Prima di descrivere il modello comportamentale è utile introdurre la convenzione di Iverson: se P è un predicato booleano che coinvolge qualche insieme di variabili, allora introducendo l'operatore [], [P] è la funzione che indica il grado di fiducia di P:

$$[P] = \begin{cases} 1, & \text{se P è vero} \\ 0, & \text{altrimenti.} \end{cases} \qquad (3.1)$$

Se assumiamo Λ come simbolo della congiunzione "AND" della logica booleana, una proprietà importante della convenzione di Iverson è che

$$[P_1 \Lambda P_2 \Lambda ... \Lambda P_n] = [P_1][P_2]...[P_n]. \qquad (3.2)$$

Quindi se un predicato P può essere scritto come congiunzione logica di operatori AND, può essere fattorizzato come in (3.1).

Prendiamo ora $x_1, \ldots, x_n$ una collezione di variabili con spazio di configurazione $S = A_1 \times \ldots \times A_n$. Con il termine di "comportamento" delle variabili si intende un qualsiasi sottoinsieme B di S e l'insieme di elementi che appartengono a B prende il nome di configurazione valida. Con questo tipo approccio un sistema viene completamente specificato dal suo comportamento B.

L'applicazione che utilizza maggiormente questo modello sono i codici. Se assumiamo come dominio di ogni variabile un alfabeto finito A, S diventa A^n e allora un sottoinsieme $C \subset S$ viene chiamato codice a blocco di lunghezza n e rappresenta il nostro comportamento, mentre le parole di codice sono la configurazione valida del codice a blocco.

La funzione caratteristica per un comportamento B è definita come $\chi_B(x_1, \ldots, x_n) \triangleq [(x_1, \ldots, x_n) \in B]$.

L'appartenenza di una configurazione a un comportamento $B \subset S$ può essere determinata applicando una serie di verifiche (*check*). Una configurazione è ritenuta valida se passa tutti i controlli. Se il predicato $(x_1, \ldots, x_n) \in B$ può essere scritto come congiunzione di predicati più semplici, allora la funzione χ_B si può fattorizzare in una serie di funzioni locali e ogni fattore indicherà quali variabili appartengono a un comportamento locale.

Spesso per semplificare la descrizione di un modello vengono introdotte delle variabili nascoste (o latenti, ausiliarie, di stato), mentre le altre variabili vengono chiamate visibili. Dato un comportamento B a cui appartengono sia variabili visibili che nascoste, esso rappresenta un comportamento C se la proiezione dei suoi elementi sulle variabili visibili è uguale a C. In questo modo qualsiasi FG che modella B è valido anche per il comportamento C. Questi tipi di FG sono stati introdotti per la prima volta da Wiberg e la classe più importante sono le rappresentazioni a traliccio (*trellis*).

3.1.1 Grafi per codici lineari

La funzione caratteristica di un codice lineare a blocco definita da una matrice H a controllo di parità $r \times n$, può essere rappresentata da un FG costruito con n nodi variabili e r nodi fattore. Ad esempio per il codice lineare binario C con matrice:

$$H = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 \end{bmatrix} \quad (3.3)$$

allora C è l'insieme delle sestuple **x** = $(x_1, \ldots, x_6)$ che soddisfa l'equazione $H\boldsymbol{x}^{\boldsymbol{T}} = 0$. La funzione caratteristica di C è

$$\chi_C(x_1, \ldots, x_6) = [(x_1, \ldots, x_6) \in C] = [x_1 \oplus x_2 \oplus x_5 = 0][x_2 \oplus x_3 \oplus x_6 = 0] \\ [x_1 \oplus x_3 \oplus x_4 = 0] \quad (3.4)$$

e il corrispondente FG di Tanner, dove i nodi fattore sono rappresentati da un simbolo speciale per il controllo di parità, è mostrato in Figura 3.1.

Una rappresentazione alternativa molto importante per un codice lineare è data dal modello a traliccio. Un *trellis* per un codice a blocco C è un grafo orientato con archi pesati dotato di radice e vertici obiettivo (*goal*), con la proprietà che ogni sequenza di archi in un cammino diretto dalla radice a un vertice *goal* è una parola di codice di C. Inoltre ogni cammino dalla radice a un vertice *goal* ha la stessa lunghezza d, che prende il nome di profondità del vertice. L'insieme dei vertici alla stessa profondità può essere visto come il dominio delle variabili di stato (nascoste) s_d. Per il codice definito da (3.4) viene proposta in Figura 3.2(a) una rappresentazione in cui i vertici alla stessa profondità sono raggruppati verticalmente, la radice è il vertice più a sinistra, mentre il vertice goal è quello più a destra.

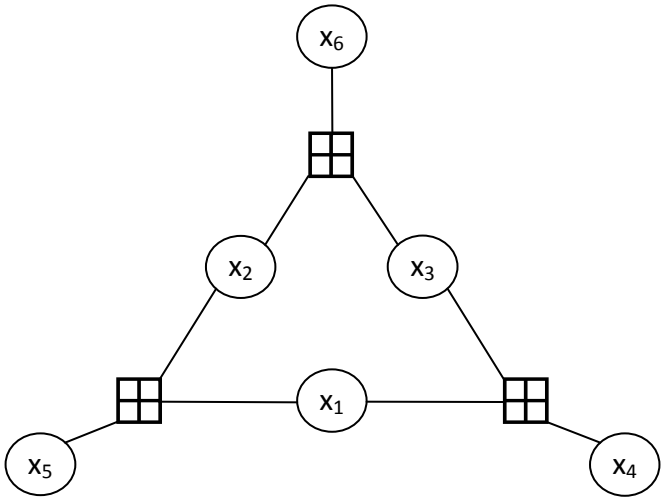

Figura 3.1. *FG di Tenner per il codice lineare a blocco (3.4).*

Globalmente un trellis definisce un comportamento nello spazio di configurazione $s_0, \ldots, s_n, x_1, \ldots, x_n$. Esso si divide in n sezioni dove la i-esima sezione T_i è un sottografo indotto dai vertici a profondità i e i-1 e costituisce un comportamento locale delle possibili combinazioni s_{i-1}, x_i, s_i. Una configurazione è valida se soddisfa tutti i vincoli locali imposti da ogni sezione. La funzione caratteristica per un comportamento si fattorizza

naturalmente in n fattori, ognuno corrispondente alla i-esima sezione con argomenti s_{i-1}, x_i, s_i.

Il corrispondente FG del trellis è mostrato in Figura 3.2(b) da cui si vede che ogni nodo fattore rappresenta una sezione del trellis.

Ad esempio, il comportamento locale T_2 consiste nelle variabili (s_1, x_2, s_2) che possono assumere i valori $s_1 = x_2 = \{0,1\}$ e $s_2 = \{0,1,2,3\}$ da cui $T_2 = \{(0,0,0), (0,1,2), (1,1,1), (1,0,3)\}$. Il corrispondente nodo fattore è la funzione indicatrice $f(s_1, x_2, s_2) = [(s_1, x_2, s_2) \in T_2]$.

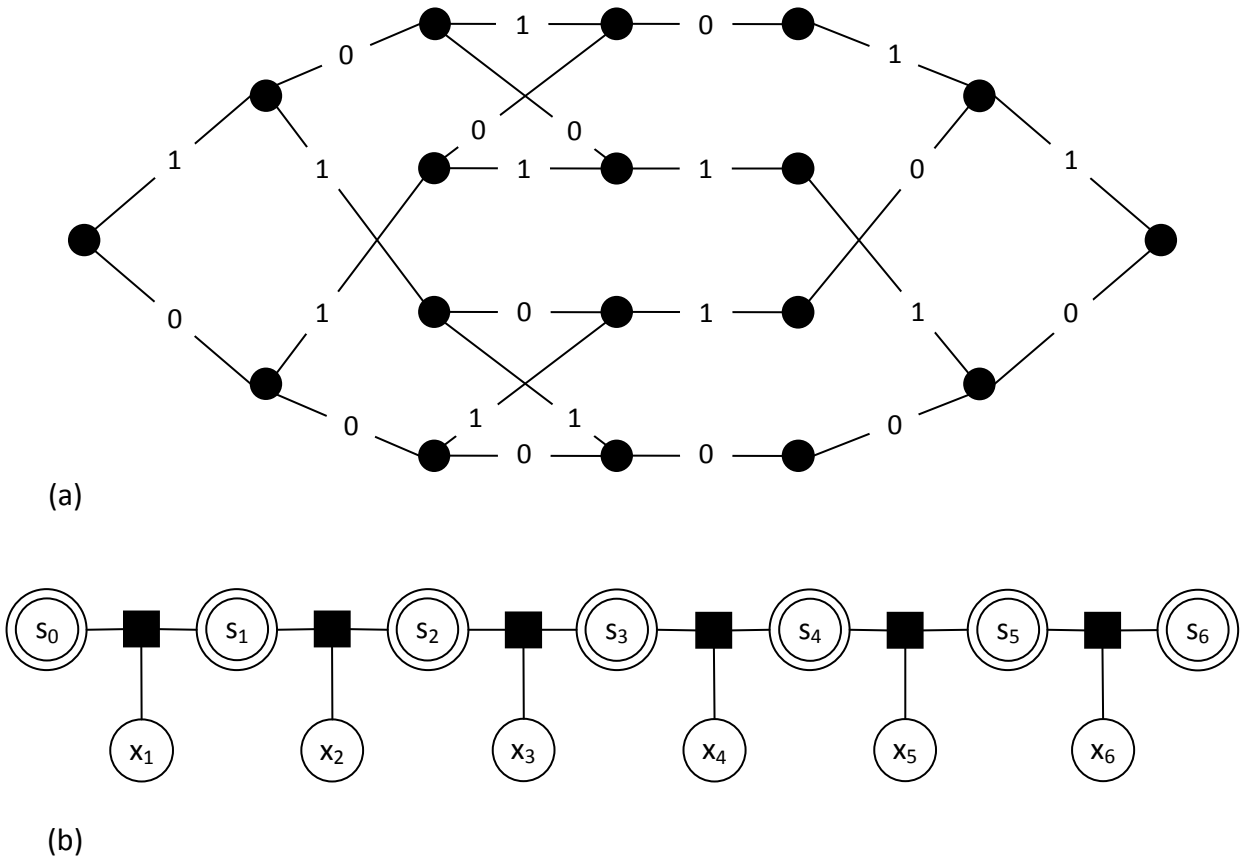

Figura 3.2 *(a) Una rappresentazione a traliccio di (3.4). (b) Il corrispondente FG.*

E' importante notare come ogni FG che corrisponde a una rappresentazione a trellis sia aciclico. Poiché ogni codice ha una rappresentazione a trellis, ogni codice può essere rappresentato con un FG aciclico. Sfortunatamente però lo spazio degli stati cresce velocemente con la complessità del codice e quindi diviene troppo grande da essere

realizzabile. L'unico modo per sopperire alla crescita degli stati è quello di introdurre cicli nel FG.

3.2 Il modello probabilistico

Un'altra importante classe di funzioni che può essere rappresentata mediante un FG sono le probabilità di distribuzione. Dalla teoria della probabilità sappiamo che l'indipendenza condizionata o non condizionata di variabili aleatorie si può esprimere come fattorizzazione delle proprie probabilità di massa congiunte o funzioni di densità di probabilità. Questa fattorizzazione, esprimibile attraverso un FG, ci può dare importanti informazioni sulla dipendenza statistica tra le variabili di un sistema.
Anche i modelli markoviani, largamente usati nelle comunicazioni e nei processi di segnali, vengono rappresentati attraverso FG con l'utilizzo di variabili nascoste poiché implicano delle fattorizzazioni non banali della probabilità di massa congiunta.

3.2.1 APP distribution

Consideriamo uno modello di codifica standard, in cui una parola di codice $x = (x_1,..,x_n)$ da trasmettere su una canale senza memoria viene selezionata da un codice C di lunghezza n. La sequenza di output corrispondente al ricevitore sarà $y = (y_1,\dots,y_n)$. Per ogni osservazione di y la distribuzione di probabilità congiunta a posteriori (APP) $p(x|y)$ è proporzionale alla funzione $g(x) = f(y|x)p(x)$, dove $p(x)$ è la probabilità *a priori* della parola di codice, mentre $f(y|x)$ è la densità di probabilità condizionata di y quando x viene trasmessa. Dato che y è fissa per ogni istante di decodifica in ricezione, il prodotto $f(y|x)p(x)$ è una funzione solo di x con i componenti di y considerati come parametri.
Assumiamo ora che la distribuzione a priori di x sia uniforme sulle parole di codice, per cui
$p(x) = \frac{\chi_C(x)}{|C|}$ dove $\chi_C(x)$ è la funzione indicatrice del codice C e $|C|$ è la cardinalità di C.
Se il canale è senza memoria, allora $f(y|x)$ si può fattorizzare come

$$f(y_1,\dots,y_n|x_1,\dots,x_n) = \prod_{i=1}^{n} f(y_i|x_i). \qquad (3.5)$$

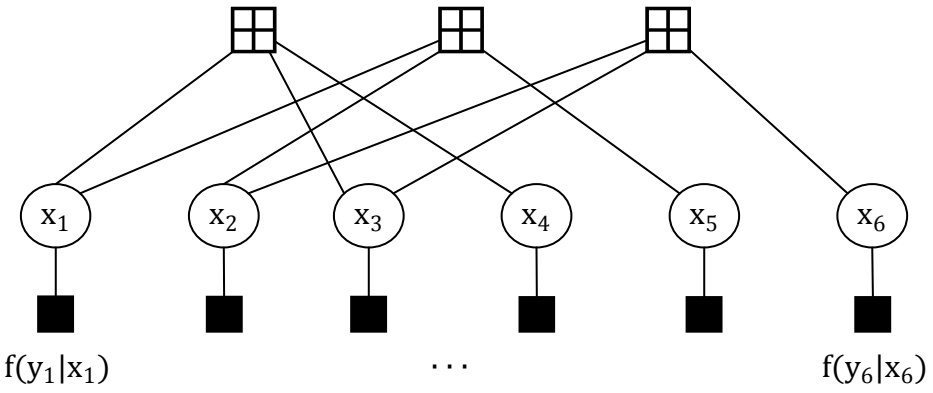

Figura 3.3. *FG per la distribuzione di probabilità a posteriori congiunta delle parole di codice date da (3.7).*

Sotto queste condizioni abbiamo

$$g(x_1, \dots, x_n) = \frac{1}{|C|}\chi_C(x_1, \dots, x_n)\prod_{i=1}^{n} f(y_i|x_i). \qquad (3.6)$$

A sua volta la funzione caratteristica $\chi_C(x)$ abbiamo visto che si può fattorizzare se soddisfa la condizione di Iverson nel prodotto di funzioni caratteristiche locali. Quindi dato un FG per $\chi_C(x)$ si può ottenere un FG per l'APP distribuition argomentando i nodi fattore con le corrispondenti densità di probabilità condizionate. Utilizzando (6) e l'esempio per il modello comportamentale abbiamo:

$$g(x_1, \dots, x_n) = [x_1 \oplus x_2 \oplus x_5 = 0][x_2 \oplus x_3 \oplus x_6 = 0][x_1 \oplus x_3 \oplus x_4 = 0] \cdot \prod_{i=1}^{n} f(y_i|x_i). \qquad (3.7)$$

Il cui FG è mostrato in Figura 3.3.

3.2.2 Hidden Markov Models

Innanzitutto per descrivere FG che modellano catene di Markov ricordiamo la regola della catena che fattorizza la distribuzione di probabilità congiunta di una collezione di variabili nel prodotto di probabilità condizionate:

$$f(x_1, \dots, x_n) = \prod_{i=1}^{n} f(x_i|x_1, \dots, x_{i-1}). \qquad (3.8)$$

Per esempio una rappresentazione di (3.8) con $n = 4$ attraverso un FG viene data in Figura 3.4(a).

Questa fattorizzazione però non da alcun vantaggio se non nel caso in cui la collezione di variabili formi una catena di Markov. Quindi (3.8) diventa assieme alla condizione di markovianità una fattorizzazione non banale:

$$f(x_1, \dots, x_n) = \prod_{i=1}^{n} f(x_i|x_{i-1}) \qquad (3.9)$$

il cui FG è mostrato in Figura 3.4(b).

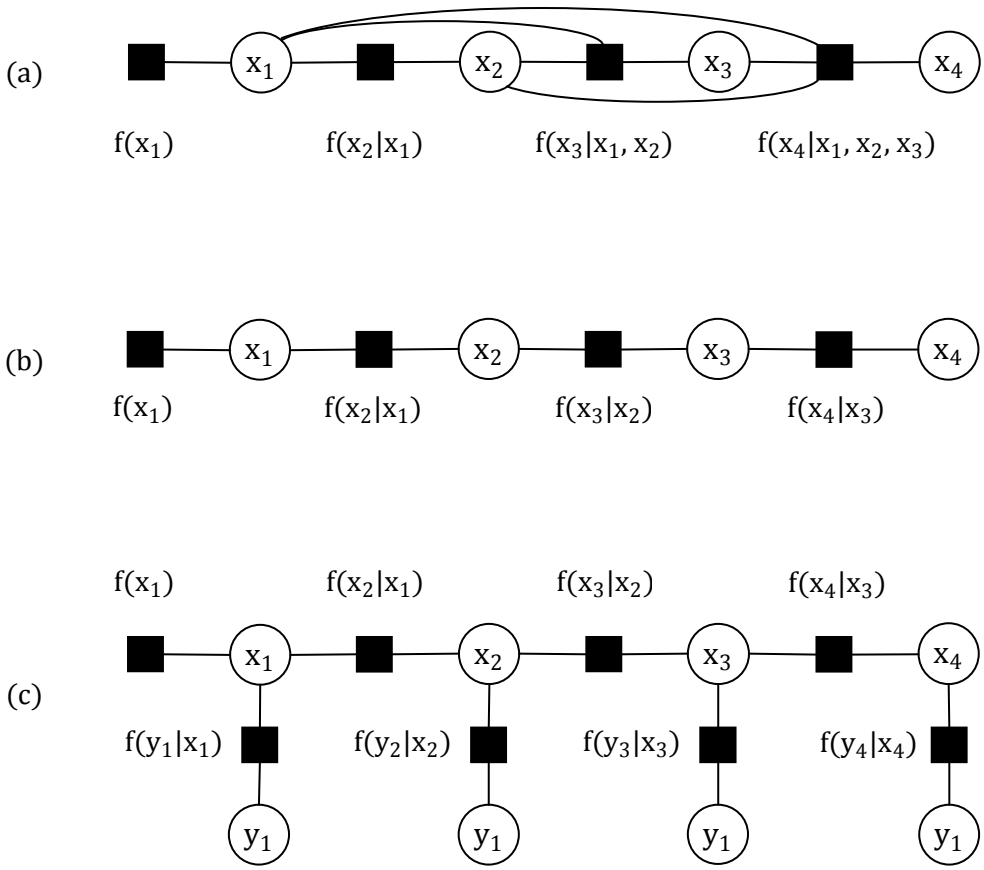

Figura 3.4. *(a) FG per la fattorizzazione della regola della catena. (b) FG per una catena di Markov. (c) FG per un modello di catena di Markov nascosta.*

Se ora supponiamo che il fenomeno di markov osservato y_i sia all'uscita di un sistema formato da un canale senza memoria con all'ingresso la collezione di variabili x_i, otteniamo un modello di markov nascosto. La probabilità di massa congiunta per le variabili di ingresso e uscita si fattorizzerà allora come:

$$f(y_1, \dots, y_n, x_1, \dots, x_n) = \prod_{i=1}^{n} f(x_i|x_{i-1}) f(y_i|x_i) \qquad (3.10)$$

la cui rappresentazione tramite FG è mostrata in Figura 3.4(c).

3.3 Forward/Backward Algorithm

Come abbiamo visto negli esempi precedenti un'importante classe di FG è quella dei grafi a catena, che rappresentano *trellis processing* e modelli di Markov. Ora applicando SPA a queste forme vedremo come il forward/backward algorithm (FBA) può essere interpretato come un suo caso particolare.

Consideriamo un caso generale in cui si ha un sistema di Markov caratterizzato da un vettore $u = (u_1, u_2, \dots, u_n)$ che rappresenta le variabili in ingresso, un vettore $x = (x_1, x_2, \dots, x_n)$ per le variabili di uscita e un vettore $s = (s_1, s_2, \dots, s_n)$ per le variabili di stato. L'uscita $x = (x_1, x_2, \dots, x_n)$ verrà poi trasmessa su un canale senza memoria e al ricevitore si potrà osservare il vettore $y = (y_1, y_2, \dots, y_n)$. Il comportamento del sistema di Markov può essere definito dalla funzione locale di controllo $T_i(s_{i-1}, x_i, u_i, s_i)$.

Come discusso in precedenza la distribuzioni a posteriori $p(u|y)$ è proporzionale a

$$g_y(u, s, x) := \prod_{i=1}^{n} T_i(s_{i-1}, x_i, u_i, s_i) \prod_{i=1}^{n} f(y_i|x_i). \qquad (3.11)$$

Il cui FG per $n = 6$ è mostrato in Figura 3.5(a). SPA applicato a questo FG calcola la distribuzione marginale APP $p(u_i|y)$ per ogni i.

Come nei FG aciclici SPA viene inizializzato dai nodi esterni, in questo caso dai nodi variabili in ingresso, dai nodi fattore $f(y_i|x_i)$ e dai nodi variabili di stato più esterni. Al passo successivo le variabili di uscita inoltreranno i messaggi in ingresso verso i corrispondenti nodi di controllo. A questo punto i controlli più esterni T_1 e T_n saranno in grado di calcolare un messaggio in uscita da inviare al nodo variabile di stato vicino. Il

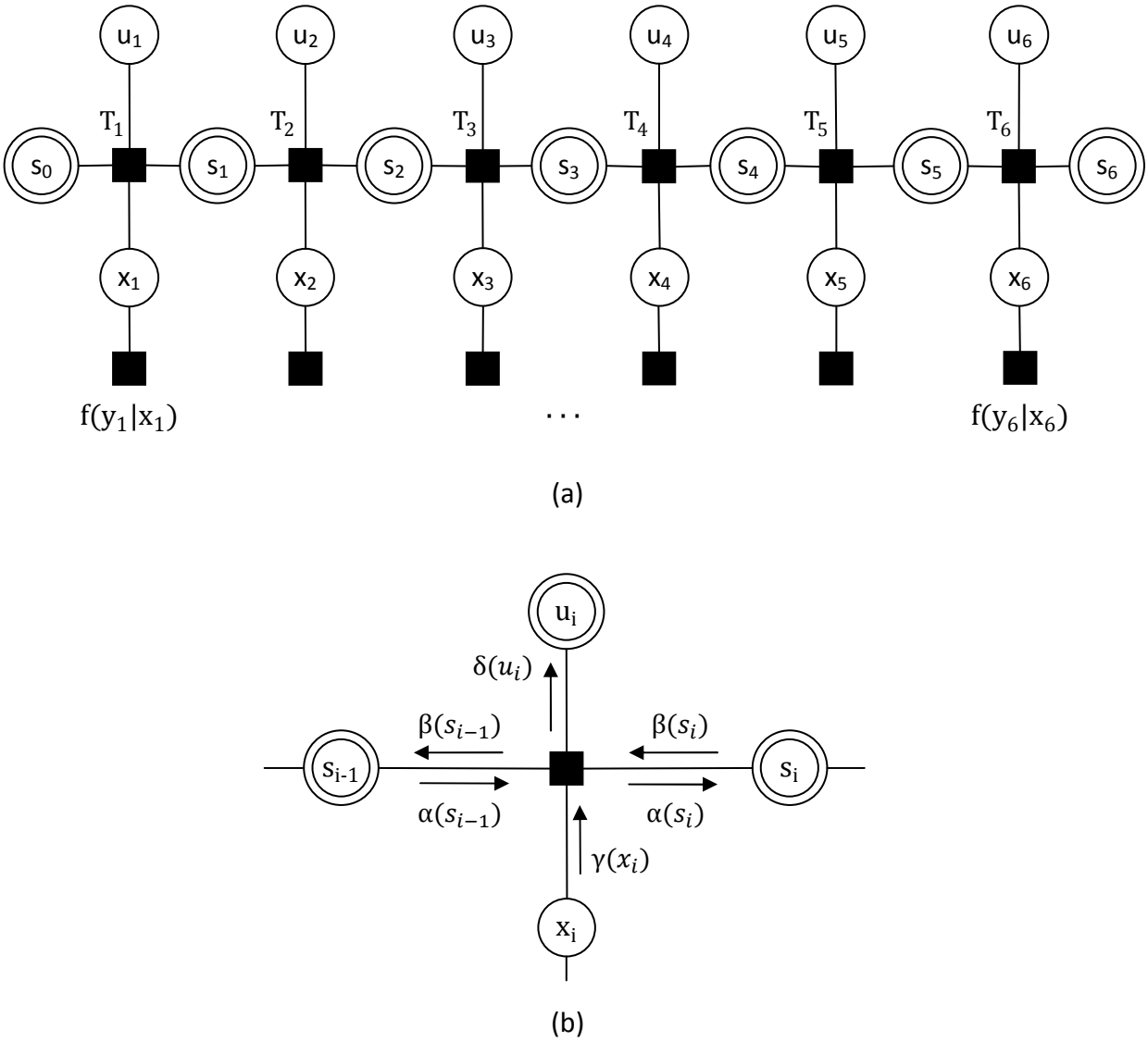

Figura 3.5. *(a) FG che rappresenta una generica combinazione di modello comportamentale e probabilistico di un sistema composto da parole di codice trasmesse su un canale senza memoria. (b) Dettaglio della sezione i-esima in cui si propagano i messaggi di BFA.*

procedimento quindi si propaga così in entrambe le direzioni fino ad arrivare agli estremi. Nella letteratura di BFA i messaggi assumono una notazione diversi da quella di SPA, in particolare $\mu_{x_i \to T_i}(x_i) = \gamma(x_i)$, $\mu_{s_i \to T_{i+1}}(s_i) = \alpha(s_i)$, $\mu_{s_i \to T_i}(s_i) = \beta(s_i)$, $\mu_{T_i \to u_i}(u) = \delta(u_i)$ la cui propagazione è mostrata per una particolare sezione nella Figura 3.5(b).

SPA genera quindi due ricorsioni per calcolare $\alpha(s_i)$ e $\beta(s_{i-1})$ che vengono chiamate forward e backward recursions. Caratterizzando le regole di aggiornamento di SPA a questo caso, otteniamo

$$\alpha(s_i) = \sum_{\sim\{s_i\}} T_i(s_{i-1}, x_i, u_i, s_i)\, \alpha(s_{i-1})\gamma(x_i) \quad (3.12)$$

$$\beta(s_{i-1}) = \sum_{\sim\{s_{i-1}\}} T_i(s_{i-1}, x_i, u_i, s_i)\, \beta(s_i)\gamma(x_i). \quad (3.13)$$

L'algoritmo termina con il calcolo dei messaggi $\delta(u_i)$

$$\delta(u_i) = \sum_{\sim\{s_{i-1}\}} T_i(s_{i-1}, x_i, u_i, s_i)\alpha(s_{i-1})\, \beta(s_{i+1})\gamma(x_i). \quad (3.14)$$

L'interpretazione che si può dare a questa somma è legata alla verifica della funzione di trellis su un arco $e = (s_{i-1}, x_i, u_i, s_i)$ tale che $T(e) = 1$.
In questo modo (3.12) e (3.13) possono essere riscritte come:

$$\alpha(s_i) = \sum_{e \in E_i(s_i)} \alpha(e)\, \gamma(e) \quad (3.15)$$

$$\beta(s_i) = \sum_{e \in E_i(s_{i-1})} \beta(e)\, \gamma(e). \quad (3.16)$$

Dove $E_i(s)$ rappresenta l'insieme degli archi incidenti su una variabile di stato s.
I messaggi α e β hanno in questo modo una interpretazione probabilistica ben definita: $\alpha(s_{i-1})$ è proporzionale alla probabilità di massa di s_{i-1} condizionata sul passato $y_1, \dots, y_{i-1}$, cioè alla probabilità condizionata che la sequenza trasmessa passi attraverso s_{i-1} conoscendo il passato; $\beta(s_i)$ è proporzionale invece alla probabilità di massa di s_i condizionata il futuro $y_{i+1}, y_{i+2}, \dots$, cioè alla probabilità condizionata che la sequenza passi attraverso gli stati s_i.

Conclusioni

I FG rappresentano uno strumento molto importante nei modelli grafici che descrivono sistemi di comunicazione. La loro prima funzione è quella di offrire una rappresentazione naturale della fattorizzazione di una funzione globale nel prodotto di funzioni locali più semplici. La generalità di codificare funzioni arbitrarie, come funzioni caratteristiche o distribuzioni di probabilità, ne fa una strumento di analisi semplice e flessibile che può essere applicato a una vasta area di applicazioni.

L'algoritmo che sfrutta meglio la visualizzazione offerta dai FG è SPA, un algoritmo di message passing da cui si possono derivare secondo una semplice regola di calcolo una grande varietà di algoritmi pratici, come FBA. SPA può venire applicato sia a FG aciclici sia a FG ciclici per calcolare le funzioni marginali associate alla funzione globale riducendo la complessità di calcolo che si avrebbe nel calcolo diretto. L'unico limite si trova nel caso ciclico, in cui la propagazione dei messaggi diventa infinita e l'algoritmo non ha una terminazione naturale. La flessibilità dei FG però garantisce grazie a delle trasformazioni di ottenere una forma aciclica partendo da un FG ciclico. In questo modo SPA può terminare calcolando le funzioni marginale esatte associate alla funzione globale. Il costo però delle trasformazioni è quello di aumentare la complessità delle funzioni locali o del dominio delle variabili riducendo così il suo potenziale in alcune applicazioni pratiche.

L'approccio che utilizzano i FG nel modellare i sistemi però è forse l'aspetto più interessante e quello che ha il maggior potenziale nello sviluppo futuro dei FG. Infatti essi possono utilizzare sia un modello comportamentale, che li rende adatti a modellare codici, sia un modello probabilistico per rappresentare sistemi di comunicazioni. I FG quindi utilizzano un approccio unificato nelle codifica e nei processi di segnali che spesso sono trattati separatamente. Per esempio un FG può unificare in un unico modello grafico la rappresentazione di un sistema formato da una sorgente e un canale di trasmissione, trattando le problematiche di stima e decodifica contemporaneamente.

Riferimenti bibliografici

1. Frank R. Kschischang, Brendan J. Frey and Hans-Andrea Loeliger (2001), "Factor Graphs and the Sum-Product Algorithm", *IEEE Transactions on Information Theory*, **47,** pp. 498 – 519.
2. Brendan J.Frey, Frank R. Kschischang, Hans-Andrea Loeliger and Niclas Wiberg (1998), "Factor Graphs and Algorithms", *Proc. 35th Allerton Conf. on Communications, Control, and Computing,* Monticello, Illinois, pp. 666 – 680.
3. Frank R. Kschischang (2003), "Codes Defined on Graphs", *IEEE Communication Magazine*, pp. 118 – 125.