Inferenza nelle reti Bayesiane

Andrea Righetti

April 1, 2022

1 Introduction

In questo progetto vogliamo replicare la tecnica del junction tree per l'inferenza probabilistica nelle reti Bayesiane. Ci limiteremo alla costruzione del junction tree come spiegato nel capitolo 4 di Jensen 1997. l'intero progetto è scritto utilizzando il linguaggio di programmazione *Python* e l'ambiente di sviluppo integrato *Pycharm* in versione 2021.2.2 (community edition).

Il calcolatore su cui è stata testata questa applicazione è un:

- *MacBook* pro retina 15' mid 2014
- CPU: intel core i7 quad-core 2.2 GHz
- RAM: 16 GB 1600MHz DDR3

Per creare un junction tree, partiamo da un normale e generico DAG (*Directed Acyclic Graph*). Nella nostra implementazione il DAG è realizzato su base di liste di adiacenza con un dizionario che ha come chiavi i vertici e come valori delle liste di vertici che rappresentano i nodi a cui il vertice in chiave è collegato.

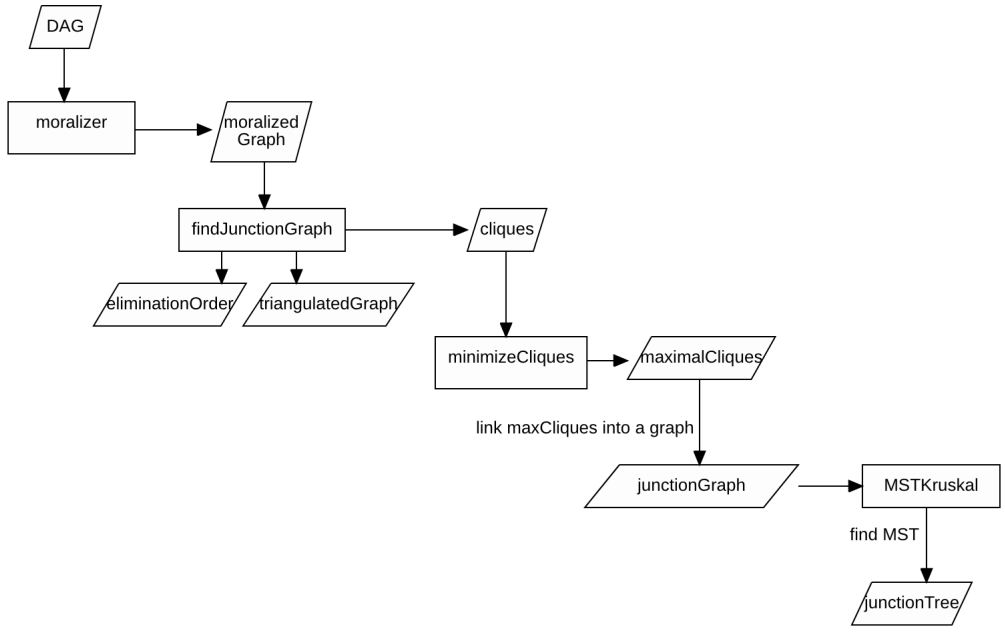


Figure 1: Flowchart di dati (parallelogrammi) e operazioni (rettangoli) che portano un DAG attraverso il processo per ricavare un junction tree.

Andiamo ad effettuare la moralizzazione su di esso, ovvero andiamo a connettere fra di loro i genitori di ogni nodo se questi non lo fossero già. Inoltre rendiamo il grafo non diretto.

Una volta moralizzato andiamo a trovare le **cricche** da cui è composto il grafo. Per trovarle dobbiamo assicurarci che il grafo sia triangolato. Per triangolare un grafo, andiamo ad elencare tutti i vertici, scegliamo un ordine di eliminazione, iniziamo ad eliminare un vertice per volta per poi collegare i vicini inducendo un grafo completo. Il grafo di partenza con unito gli archi aggiunti durante il processo di eliminazione dei nodi, chiamati *fillins*, forma il **grafo triangolato**.

Il grafo triangolato così ottenuto non è univoco, dipende dall'ordine di eliminazione. Ogni ordine di eliminazione è corretto, ma l'ordine ottimale dovrebbe aggiungere meno fillins possibili. Per ottenere ciò andiamo a ripetere l'operazione di triangolazione 100 volte, segnandoci ogni volta il numero di *fillins*, andiamo poi a scegliere l'ordine di eliminazione che ha prodotto il minor numero di *fillins*.

I nodi rimossi nell'ordine di eliminazione insieme ai loro vicini formano le varie cricche.

Trovate le cricche troviamo le **cricche massimali**, ovvero le cricche che non sono contenute in altre cricche.

Connettiamo le cricche massimali in un grafo, quello che otteniamo è chiamato **junction graph**. Nel nostro caso il junction graph è individuato da una matrice di adiacenza.

Applichiamo l'algoritmo di *Kruskal* per trovare il **Maximum Spanning Tree**. L'albero ricavato sarà finalmente il **junction tree**.

2 risultati

2.1 DAG in figura 4.17

Come da richiesta abbiamo applicato l'algoritmo sopra descritto all'esercizio 4.18 che fa capo al DAG presente in figura 4.17 del libro di Jensen 1997.

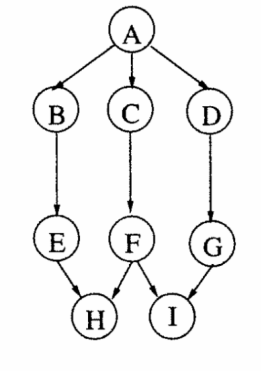
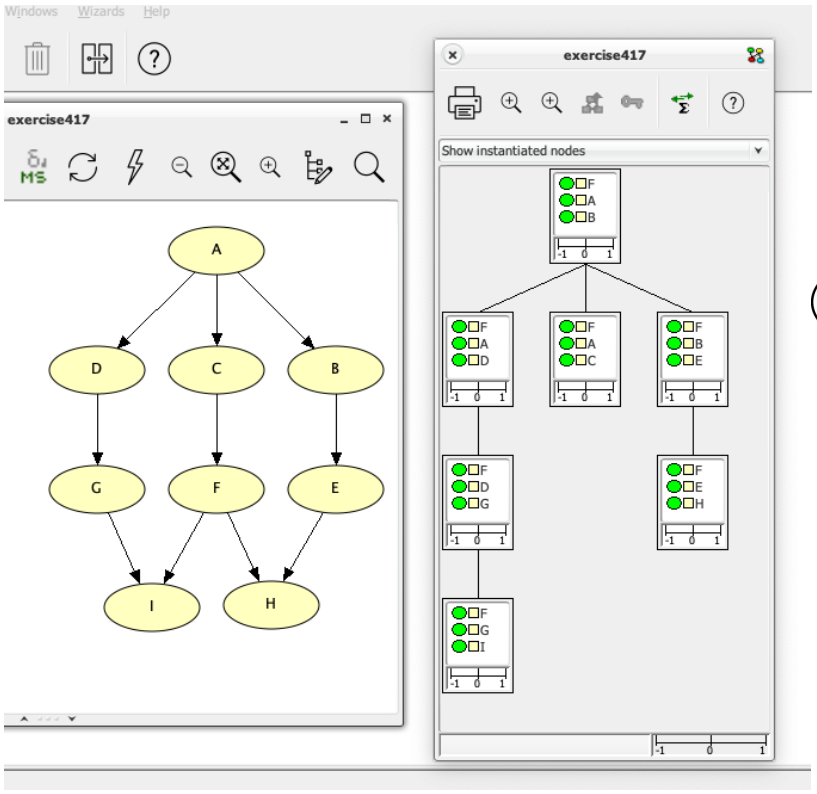


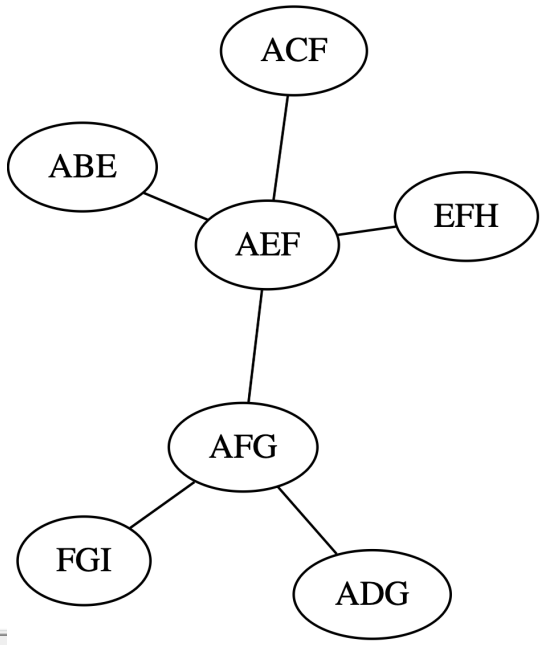
Figure 2: DAG esercizio 4.17

Il junction tree ottenuto dal nostro applicativo è quello mostrato in figura 3b lo compariamo al junction tree ottenuto dal programma Hugin lite (figura 3a).

Possiamo notare che i risultati mostrati differiscono, questo si può spiegare considerando che l'albero triangolato è differente a seconda dell'ordine di eliminazione dei nodi. Il grafo di partenza ha 9 nodi ciò significa 9! (=362880) permutazioni diverse del vettore dei nodi per l'eliminazione. Noi ne abbiamo provate solo 100 quindi è altamente probabile che l'ordine di eliminazione non sia il medesimo. Questo spiegherebbe il risultato diverso ottenuto.



(a) junction tree secondo hugin lite



(b) junction tree secondo il nostro programma

Figure 3: junction tree per il DAG in figura 4.17 (Jensen 1997)

2.2 DAG a piacere

Per quanto riguarda il DAG scelto a piacere ho scelto quello in figura 4. I risultati sono quelli scelti in figura 5

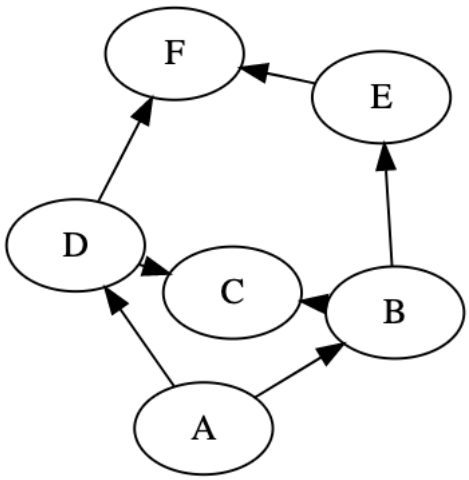
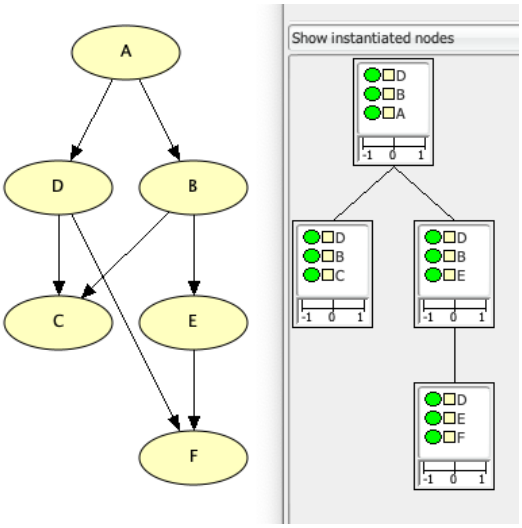
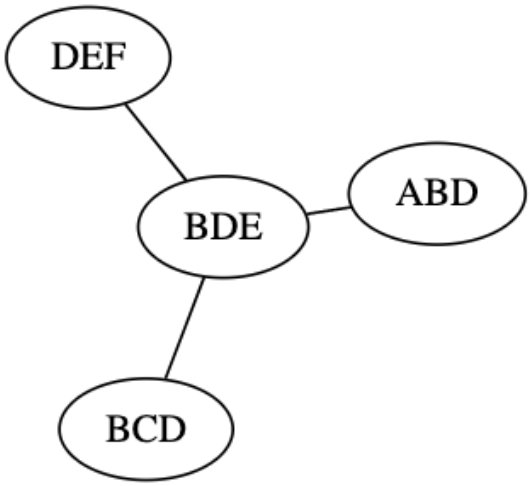


Figure 4: DAG a piacere



(a) junction tree secondo hugin lite



(b) junction tree secondo il nostro programma

Figure 5: junction tree per il DAG a piacere di figura 4 in figura 4.17 (Jensen 1997)