Progetto Incertezza

Altamura Sante Mat. 941238
Mazzone Giuseppe Mat. 938528

July 2020

Contents

1 Introduzione

L'obiettivo dei due progetti consiste nell'eseguire diversi esperimenti su Reti Bayesiane (BN) e Reti Bayesiane Dinamiche (DBN) al variare di:

- **topologia** delle rete
- tipologia di **pre-processing** adottata
- **numero** di variabili di query e di evidenza

al fine di confrontare i **tempi di esecuzione** impiegati e valutare criticamente i risultati, sulla base delle assunzioni fatte in partenza.

2 Progetto su Bayesian Networks (BN)

Al fine di eseguire diversi esperimenti su molte BN diverse, è stata implementata una struttura di classi JAVA per l'elaborazione delle reti in base al tipo di preprocessing adottato, in grado di fornire un supporto per la creazione di sottografi, utili per ricavare alcune informazioni dalla rete iniziale.

2.1 Implementazione in JAVA

Il progetto si articola secondo le seguenti classi:

- **PreprocessingInfo.java**:

  Classe utilizzata per tener traccia di:

  - Variabili irrilevanti
  - Archi irrilevanti
  - Min Degree Order e Min Fill Order
  - Archi della rete
  - Evidenze della query

- **InteractionGraph.java**

  Necessaria per la costruzione dell'Interaction Graph, utile al fine di stabilire il Min Degree Order o il Min Fill Order de utilizzare nella fase di Variable Elimination (VE).

- **MoralGraph.java**

  Necessaria per la costruzione dell'Moral Graph, utile al fine di effettuare i test di M-separation tra le variabili della BN e le variabili di query.

- **ComplexityAnalyzer.java**

  È la classe principale e contiene i metodi per la creazione della BN.
  Metodi più importanti:

  - **ComplexityAnalyzer** (costruttore) si occupa di calcolare la distribuzione a posteriori in base a:

    1. tipologia di pruning adottata (Node Pruning, M-Separation, Edge Pruning)

    2. ordine euristico (Min Degree Order, Min Fill Order, Ordine Topologico Inverso) con cui la VE processerà le variabili

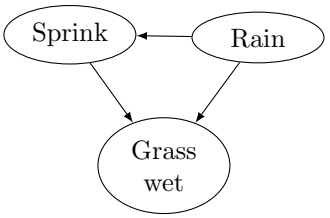
- **modifyBN** si occupa della modifica della BN, costruita con la libreria AIMA-CORE, a seguito degli algoritmi di pruning.

  1. Node Pruning: consiste nella semplice rimozione dei nodi irrilevanti dalla rete, con tutti i loro archi entranti e uscenti.

  2. M-Separation: rimozione dei nodi m-separati dalla query attraverso un'evidenza.

  3. Edge Pruning: è il caso più interessante e consiste nella rimozione degli archi uscenti da un nodo di evidenza con conseguente modifica delle Conditional Probability Table (CPT) dei nodi che hanno perso i padri.
     esempio con evidenza Sprink = T

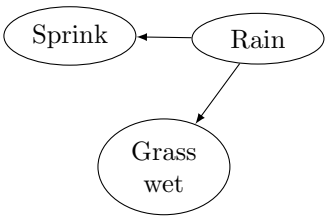
	Sprink	
Rain	T	F
F	0.4	0.6
T	0.01	0.99



Rain	
T	F
0.2	0.8

		Grass wet	
Sprink	rain	T	F
F	F	0.4	0.6
F	T	0.01	0.99
T	F	0.01	0.99
T	T	0.01	0.99

	Sprink	
Rain	T	F
F	0.4	0.6
T	0.01	0.99



Rain	
T	F
0.2	0.8

	Grass wet	
Rain	T	F
F	0.01	0.99
T	0.01	0.99

Consideriamo i dominii dei padri di Grass Wet:

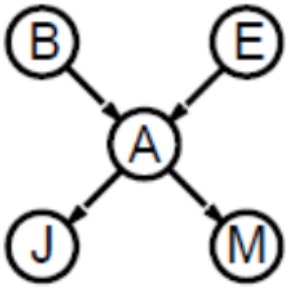
Sprink = [T], Rain = [T,F]
Prodotto Cartesiano = [T,T] [T,F]

Quindi andiamo a considerare nella CPT di Grass Wet, i valori per ogni combinazione del Prodotto Cartesiano, ed eliminiamo la variabile Sprink dalla stessa, ottenendo una nuova CPT per Grass Wet.

- **EliminationDarwicheAsk.java**

  Permette di applicare l'algoritmo di Variable Elimination (VE) su un qualsiasi ordinamento datogli in input.

  Esempio: Se consideriamo l'ordinamento π = [E,M,J,A,B], quando la VE



processa la variabile E, questa non ha ancora processato i suoi figli e di conseguenza non si dispone di tutti i fattori in cui E compare.
La VE secondo Darwiche "recupera" e aggiunge all'insieme dei fattori anche i fattori che menzionano la variabile processata (in questo caso E); fatto questo è possibile eliminarla.

2.2 Sperimentazione sulle BN

Sono state eseguite diverse sperimentazioni sulla rete, confrontando il tempo di esecuzione al variare di:

- Topologia della rete:
  - Dimensione → numero di nodi
  - Complessità → "lontananza" da un polytree

    Tutte le reti utilizzate per la sperimentazione non rappresentano un polytree, in quanto eliminando la direzionalità agli archi della rete, si ottiene più di un ciclo, soprattuto per reti con un grande quantitativo di nodi.

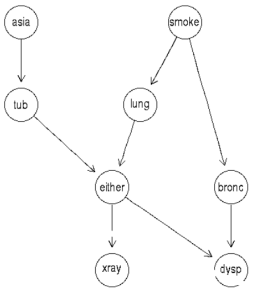
    Questo è dovuto al fatto che due nodi della rete possono essere collegati da più di un cammino. Di conseguenza, all'aumentare della complessità della rete, cioè all' "allontarasi" da un polytree, aumenta anche la treewidth, cioè la minima width di tutti gli ordinamenti possibili delle variabili di una rete.

    Man mano che aumenta la complessità di una rete, aumenta la sua treewidth. Poichè non si hanno a disposizione tutti gli ordinamenti possibili della rete, ciò che è possibile confrontare sono le width degli ordinamenti calcolati secondo determinate euristiche.
- Numero e posizione delle variabili di query e di evidenza
- Ordine delle variabili
  - Min Degree Order
  - Min Fill Order
  - Ordine Topologico Inverso

  I test secondo l'ordine delle variabili sono stati effettuati utilizzando una variabile di query e una di evidenza rispettivamente scelte tra le foglie della rete e i nodi intermedi.

2.2.1 Reti piccole (meno di 20 nodi)

- Asia:
  - Numero nodi: 8
  - Numero archi: 8
  - Maximum in-degree: 2
  - Average Degree: 1.00
  - Width: 2
  - Query: P(Xray | Lung = Yes)



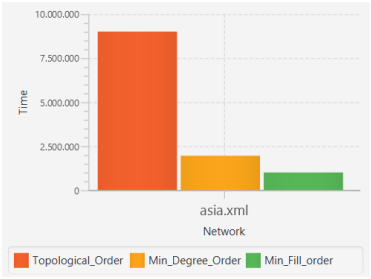
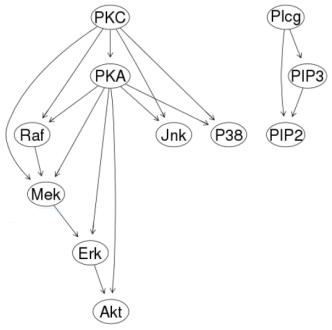


Figure 1: Tempo (nano secondi)

- Sachs:
  - Numero nodi: 11
  - Numero archi: 17
  - Maximum in-degree: 3
  - Average Degree: 1.54
  - Width: 4
  - Query: P(Akt | Raf = LOW)



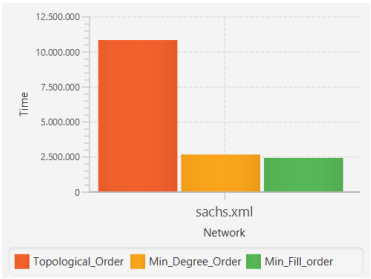
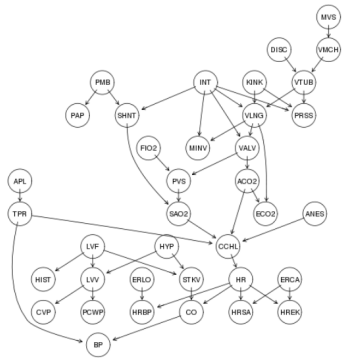


Figure 2: Tempo (nano secondi)

2.2.2 Reti medie (20 - 50 nodi)

- Alarm:
  - Numero nodi: 37
  - Numero archi: 46
  - Maximum in-degree: 4
  - Average Degree: 1.24



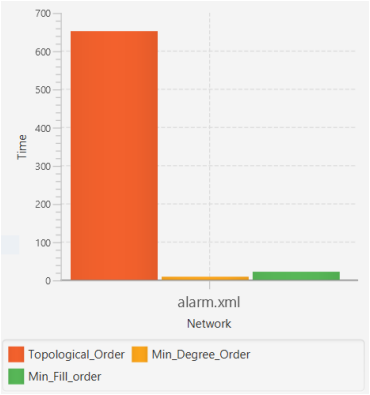


Figure 3: Tempo (millisecondi)

  - Width:
    a) Inverse Topological Order: 12
    b) Min Deegree Order: 4
    c) Min Fill Order: 4

Dal grafico in Figura 3 si inizia a notare la differenza in termini di tempo tra l'esecuzione della VE secondo l'ordine topologico inverso e secondo gli ordini basati sulle euristiche.

Questa differenza è dovuta sia al modo in cui sono processate le variabili della rete, sia alla width degli ordinamenti: nell'ordine topologico inverso il numero di variabili del fattore più grande è tre volte quello degli altri due ordinamenti.

- Insurance:
  - Numero nodi: 27
  - Numero archi: 52
  - Maximum in-degree: 3
  - Average Degree: 1.93

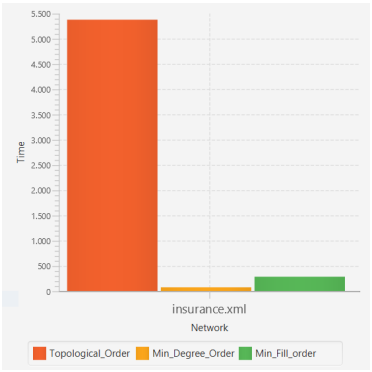
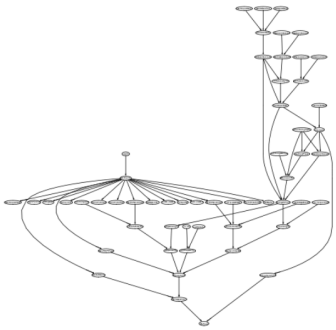


Figure 4: Tempo (millisecondi)

  - Width:
    a) Inverse Topological Order: 11
    b) Min Deegree Order: 7
    c) Min Fill Order: 8

2.2.3 Reti grandi (50 - 100 nodi)

- HailFinders:
  - Numero nodi: 56
  - Numero archi: 66
  - Maximum in-degree: 4
  - Average Degree: 1.17



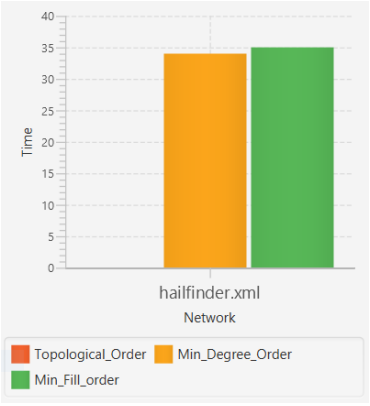


Figure 5: Tempo (millisecondi)

  - Width:
    a) Inverse Topological Order: 12 (a runtime)
    b) Min Deegree Order: 5
    c) Min Fill Order: 5

    In questo caso non è stato possibile inserire il tempo di esecuzione della VE secondo **l'ordine topologico inverso** in quanto è stato raggiunto il **limite di memoria** per l'esecuzione del programma.

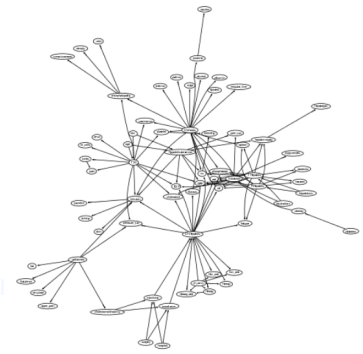
    Quindi, ha senso chiedersi la complessità temporale per l'ordine topologico inverso. Tenendo conto che:

    * è stata formulata una **query** in cui si chiede la distribuzione di probabilità a posteriori di una **variabile foglia**
    * con **evidenza** rappresentata da un **nodo intermedio**
    * la **width** calcolata a runtime è w = 12
    * la **cardinalità del dominio** più grande è d = 11

    la **complessità temporale** è $O(d^w n) = O(11^{12}56)$.

E' inevitabile notare come il tempo di esecuzione della VE si riduca da incalcolabile a questione di pochi millisecondi solamente cambiando l'ordine con cui processare le variabili.
Questo fa si che la width dell'ordinamento cali drasticamente.

- Hepar2:
  - Numero nodi: 70
  - Numero archi: 123
  - Maximum in-degree: 7
  - Average Degree: 1.73



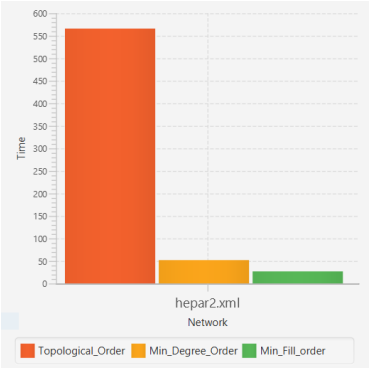


Figure 6: Tempo (millisecondi)

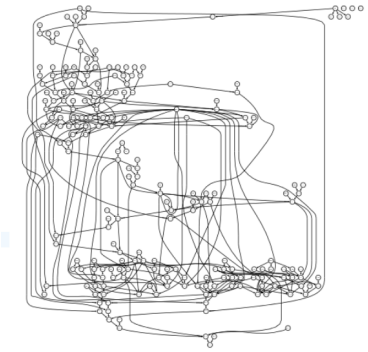
  - Width:
    a) Inverse Topological Order: 13
    b) Min Deegree Order: 7
    c) Min Fill Order: 7

Notiamo che, nonostante questa rete abbia più nodi e più archi della precedente, la complessità temporale è notevolmente ridotta perchè la cardinalità del dominio massimo è solo d = 4.
Di conseguenza risulta molto meno dispendioso il calcolo della VE.

Complessità temporale(ITO) è $O(d^w n) = O(4^{13}70)$
Complessità temporale(MDO) è $O(d^w n) = O(4^7 70)$
Complessità temporale(MFO) è $O(d^w n) = O(4^7 70)$

2.2.4 Reti molto grandi (100 - 1000 nodi)

- Andes:
  - Numero nodi: 223
  - Numero archi: 338
  - Maximum in-degree: 6
  - Average Degree: 1.51



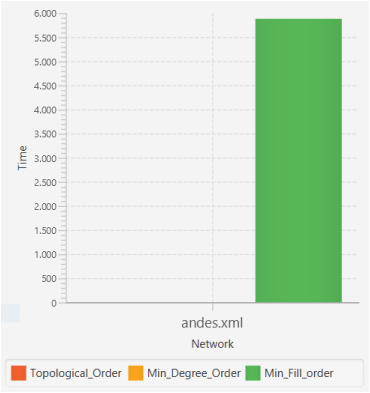


Figure 7: Tempo (millisecondi)

  - Width:
    a) Inverse Topological Order: 20 (a runtime)
    b) Min Deegree Order: 16 (a runtime)
    c) Min Fill Order: 18

Per questa rete non si è riusciti a calcolare la distribuzione a posteriori della variabile di query sia nel caso di VE con ordine topologico inverso che nel caso di VE con Min Deegre Order.

Notiamo che, nonostante la cardinalità del dominio sia solo d = 2, le width dei primi due ordinamenti sono molto alte. Tuttavia, la VE che processa le variabili secondo il Min Fill Order riesce ad ottenere una soluzione in poco più di 5 secondi.

Complessità temporale(MFO): $O(d^w n) = O(2^{18}223)$

2.2.5 Valutazione pruning rispetto alla posizione della query e dell'evidenza e al loro numero

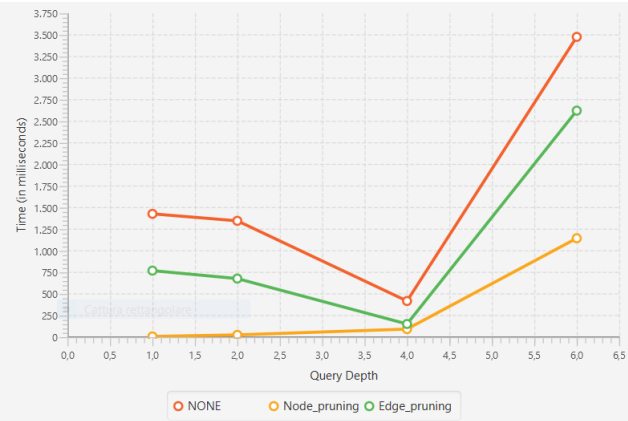
Lo scopo adesso, è quello di valutare, per ogni algoritmo di pruning, il tempo impiegato dalla VE per calcolare la distribuzione.
Prendiamo come esempio una BN di medie dimensioni e una di grandi dimensioni:

- Insurance: 27 nodi

1) fissata una evidenza, calcolare il tempo di VE per una query, variando la sua profondità nella BN.

   Esempio:
   - Evidenza $\longrightarrow$ Age = Adult
   - Profondità = 1 $\longrightarrow$ Query = GoodStudent
   - Profondità = 2 $\longrightarrow$ Query = SeniorTrain
   - Profondità = 4 $\longrightarrow$ Query = Accident
   - Profondità = 6 $\longrightarrow$ Query = PropCost

   Risultati:

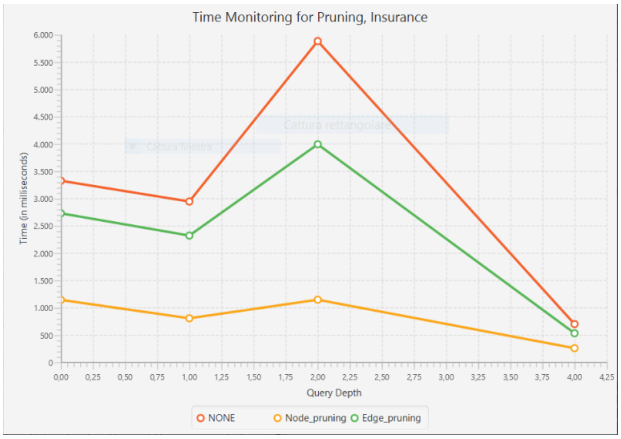


2) fissata una query, calcolare il tempo di VE, variando la profondità di una evidenza.

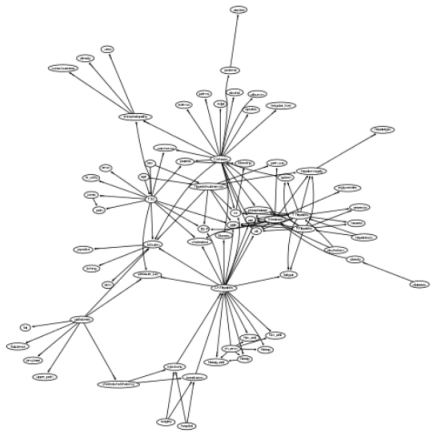
Esempio:

* Query $\longrightarrow$ PropCost
* Profondità $= 0 \longrightarrow$ P(PropCost$|$ Age = Adult)
* Profondità $= 1 \longrightarrow$ P(PropCost$|$ RiskAversion = Normal)
* Profondità $= 2 \longrightarrow$ P(PropCost$|$ MakeModel = SportsCar)
* Profondità $= 4 \longrightarrow$ P(PropCost$|$ Accident = Moderate)

Risultati:



- Hepar2: 70 nodi

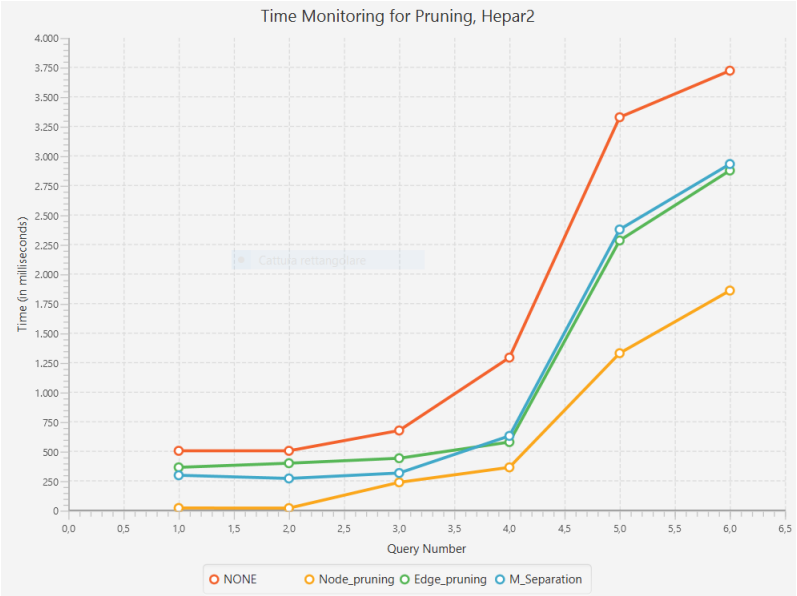


1) fissata una evidenza, calcolare il tempo di VE, variando il numero di query.
Esempio:

– Evidenza $\longrightarrow$ Obesity = Present
– Numero Query $= 1 \longrightarrow$ P(Alcohol$|$ Obesity = Present)

- Numero Query = 2 $\longrightarrow$ P(Alcohol, Nausea| Obesity = Present)
- Numero Query = 3 $\longrightarrow$ P(Alcohol, Nausea, Hepatalgia| Obesity = Present)
- Numero Query = 4 $\longrightarrow$ P(Alcohol, Nausea, Hepatalgia, Proteins| Obesity = Present)
- Numero Query = 5 $\longrightarrow$ P(Alcohol, Nausea, Hepatalgia, Proteins, Platelet| Obesity = Present)
- Numero Query = 6 $\longrightarrow$ P(Alcohol, Nausea, Hepatalgia, Proteins, Platelet, Hbsag| Obesity = Present)

Risultati:



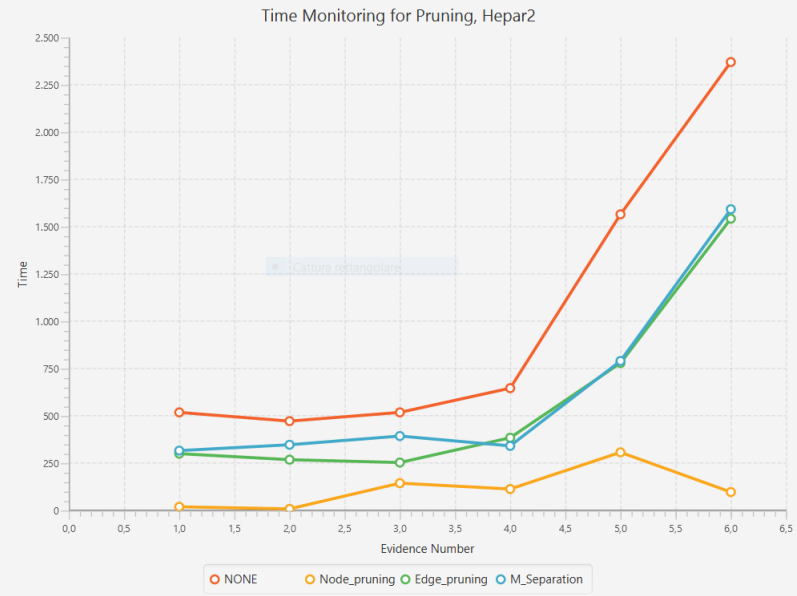
Valori della width al variare del numero di query

Query Number	NONE	Node Pruning	Edge Pruning	M Separation
1	12	7	12	12
2	12	7	12	12
3	13	12	13	13
4	14	13	14	14
5	15	14	15	15
6	16	15	16	16

2) utilizzando le query precedenti, variare il numero di evidenze.
Esempio:

- Numero Evidenze = 1 $\longrightarrow$ P(Alcohol| Obesity = Present)
- Numero Evidenze = 2 $\longrightarrow$ P(Alcohol, Nausea| Obesity = Present, Hepatotoxic = Present)
- Numero Evidenze = 3 $\longrightarrow$ P(Alcohol, Nausea, Hepatalgia| Obesity = Present,Hepatotoxic = Present, Hospital = Present)
- Numero Evidenze = 4 $\longrightarrow$ P(Alcohol, Nausea, Hepatalgia, Proteins| Obesity = Present,Hepatotoxic = Present, Hospital = Present, Surgery = Present)
- Numero Evidenze = 5 $\longrightarrow$ P(Alcohol, Nausea, Hepatalgia, Proteins, Platelet| Obesity = Present,Hepatotoxic = Present, Hospital = Present, Surgery = Present, Diabetes = Present)
- Numero Evidenze = 6 $\longrightarrow$ P(Alcohol, Nausea, Hepatalgia, Proteins, Platelet, Hbsag| Obesity = Present,Hepatotoxic = Present, Hospital = Present, Surgery = Present, Diabetes = Present, Fibrosis = Present)

Risultati:



Valori della width al variare del numero di evidenze

Evidence Number	NONE	Node Pruning	Edge Pruning	M Separation
1	12	7	12	12
2	12	7	12	12
3	12	10	12	13
4	12	10	12	12
5	13	12	13	13
6	14	10	14	14

Da questi ultimi due esperimenti, è possibile notare che:

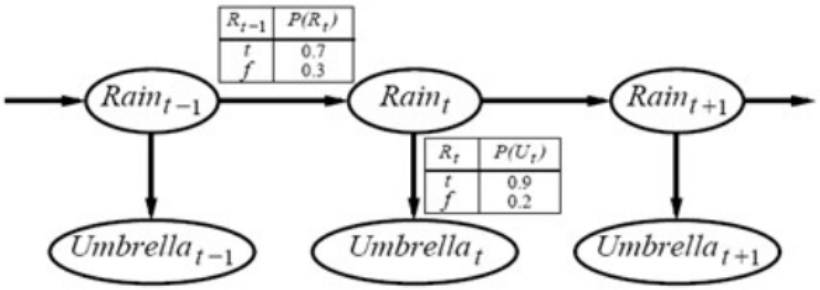
- Al aumentare del numero di query aumenta il tempo di calcolo della VE;
- Il Node Pruning si dimostra la tecnica migliore per ridurre il costo computazionale;
- All'aumentare del numero di evidenze diminuisce il tempo di calcolo della VE;
- L'aumentare del numero di evidenze fortifica l'effetto dell'Edge Pruning;
- Dalle tabelle è possibile notare la variazione della width in relazione al grafico;

3 Progetto su Dynamic Bayesian Networks (DBN)

Le DBN modellano le relazioni temporali tra le variabili. Mentre una BN rappresenta solo relazioni probabilistiche tra un insieme di variabili in un determinato momento, un DBN mette in relazione il valore di alcune variabili con il suo valore in punti precedenti e con i valori di altre variabili in punti precedenti. Al fine di operare con reti Bayesiane dinamiche bisogna tener conto di tre principali componenti:

- $P(X_0) \longrightarrow$ Probabilità a priori.
- $P(X_t \mid X_{t+1}) \longrightarrow$ Modello di transiozione.
- $P(E_t \mid X_{t)} \longrightarrow$ Modello sensoriale.

Una possibile rete dinamica potrebbe essere rappresentata nel seguente modo, dove vediamo come variabile di stato `Rain` (di transizione) e come variabile di evidenza `Umbrella` (sensoriale):



3.1 Variable Elimination per DBN: Rollup Filtering

Partendo dalla probabilità ai parametri precedentemente specificati è possibile costruire un numero illimitato di intervalli della rete dinamica, copiando il primo intervallo; questa operazione si chiama `unrolling`. Tuttavia tale tecnica, soffre di un'eccessiva occupazione della memoria, poiché tutti i passaggi nel tempo (che tendono all'infinito) sono mantenuti in memoria.

Usiamo la tecnica del `Rollup Filtering` per risolvere questo problema; essa rende possibile concentrarsi su due "slice" per volta. Data una query, e una eventuale sequenza di osservazioni, l'algoritmo calcola prima di tutto la VE sul primo slice ovvero quello in cui abbiamo la probabilità a priori delle variabili di transizione X_0 e eventuali evidenze per le variabili allo stato X_1; il risultato di questa VE sarà utilizzato per il calcolo della VE allo slice successivo. In questo modo, attrverso una chiamata ricorsiva dell'algoritmo della VE su ogni slice è possibile effettuare il filtering.

3.2 Implementazione in Java

Per l'implementazione in Java si è utilizzato gran parte del codice prodotto nella prima esercitazione in riferimento sia all'algoritmo di Variable Elimination che

agli algoritmi di ordinamento delle variabili. Implementazioni aggiuntive sono le seguenti classi:

- **RollupFiltering.java**:

  Classe utilizzata per implementare l'algoritmo di rollup su detto, esso prevede la modifica della VE in modo tale che venga eseguita a slice. Nello specifico troviamo funzioni quali:

  - getQueryTm1: data una variabile di query è in grado di riconoscere la variabile di stato precedente sul quale effettuare ricorsivamente la VE.
  - getVariablesQuery & getAttualQuery: permettono di ottenere le variabili di query e le eventuali evidenze che fanno riferimento allo slice specifico di riferimento.
  - getRVtoEliminateForSlice: permette di identificare tutte le Random Variable della rete che non fanno riferimento allo slice considerato.
  - getCommonNode: permette di identificare l'insieme di nodi che sono sia fra gli ancestor dell'evidenza che tra i successor della variabile di query.
  - calculateVariablesRollup: permette di identificare l'insieme dei nodi che fanno riferimento allo slice attuale (e quindi alla query attuale) andando a eliminare dall'insieme totale dei nodi tutte le variabili prodotte dalla funzione precedente. Essa inoltre restituisce anche l'insieme di variabili che risultano essere hidden.
  - EliminationAtSlice: leggera modifica dell'algoritmo di VE presentato nella sezione precedente in cui si considerano i risultati dello slice precedente per poter effettuare il calcolo dello slice successivo, tale implementazione è resa possibile grazie ad un parametro che viene passato ricorsivamente.

- **ComplexDBN.java**

  È la classe che implementa 4 diverse reti dinamiche con diverso numero di variabili di transizione, diverse evidenze, e diversi nodi hidden. Varia anche il numero di stati temporali.

- **ComplexexityAnalyzer2.java**

  È la classe principale che contiene la definizione delle varie reti con annesse query ed evidenze, e il main con le varie combinazioni di esecuzione al varare di:

  1. tipologia di rete (più o meno variabili di stato e più o meno evidenze).

2. tipologia di query (più o meno variabili di stato e più o meno evidenze).
3. ordine euristico (Min Degree Order, Min Fill Order, Ordine Topologico Inverso) con cui la VE processerà le variabili di ogni slice.

Inoltre vengono monitorati i tempi di elaborazione della VE ad ogni slice e il tempo di ordinamento delle varabili .

- **GraphicalRepresentation.java**

  In questa classe, a partire dalle strutture dati create nella classe precedente, avviene la realizzazione di grafici.

3.3 Sperimentazione sulle DBN

Sono state eseguite diverse sperimentazioni, confrontando il tempo di esecuzione al variare di:

- Topologia della rete:
  - Dimensione → numero di totale di nodi
  - Complessità delle variabili di transizione → numero delle variabili di transizione e complessità della relazione tra stati temporali
  - Complessità delle modello sensoriale → numero delle variabili di evidenza
- Ordine delle variabili nel processo di VE
  - Min Degree Order
  - Min Fill Order
  - Ordine Topologico Inverso
- Query
  - Su più o meno variabili di stato e con più o meno evidenze
  - numero dimensione massima fattore intermedio

3.3.1 Reti utilizzate

In primo luogo si è utilizzata la rete vista a lezione sulla pioggia data l'evidenza dell'ombrello. Si sono creati più slice temporali e le variabili considerate dalla VE nel rollup filtering sono quelli rappresentati nella seguente figura:

Come ulteriore evoluzione della rete Rain si è aggiunta una ulteriore variabile di transizione, Wind, che è direttamente collegata allo stato Wind precedente e anche allo stato Rain; si sono creati 10 passaggi di stato e si è posta una query di esempio su Rain e Wind date le evidenze a falso per l'ombrello.

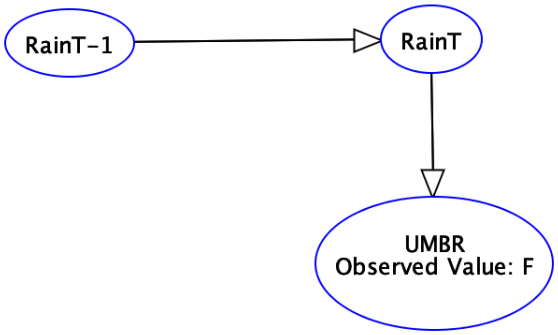


Figure 8: Rappresentazione di uno slice nella DBN Rain

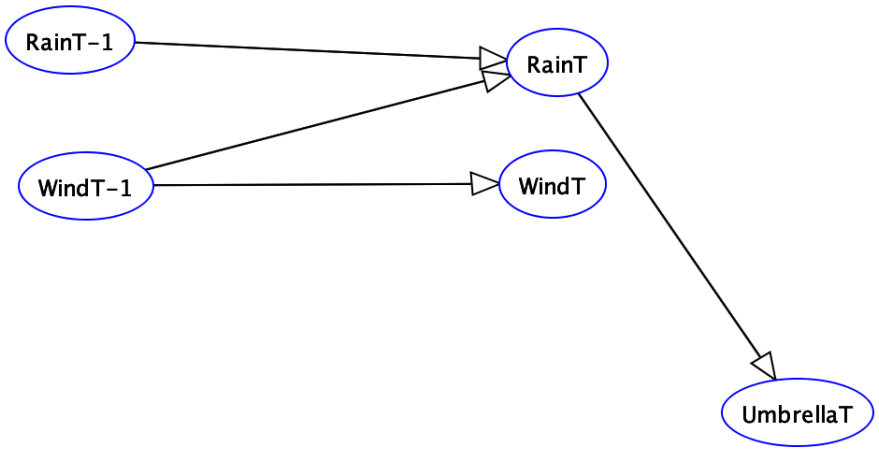


Figure 9: Rappresentzione di uno slice nella DBN Rain-Wind

Una ulteriore rete testata risulta essere una dinamizzazione della rete Alarm vista nelle BN statiche. Si è deciso di rendere come variabili di transizione Burglary e Heartquake e di variare le evidenze tra Alarm, John Calls e Mary Calls. Si è posta la query sulle 2 variabili di stato, andando a variare le evidenze.

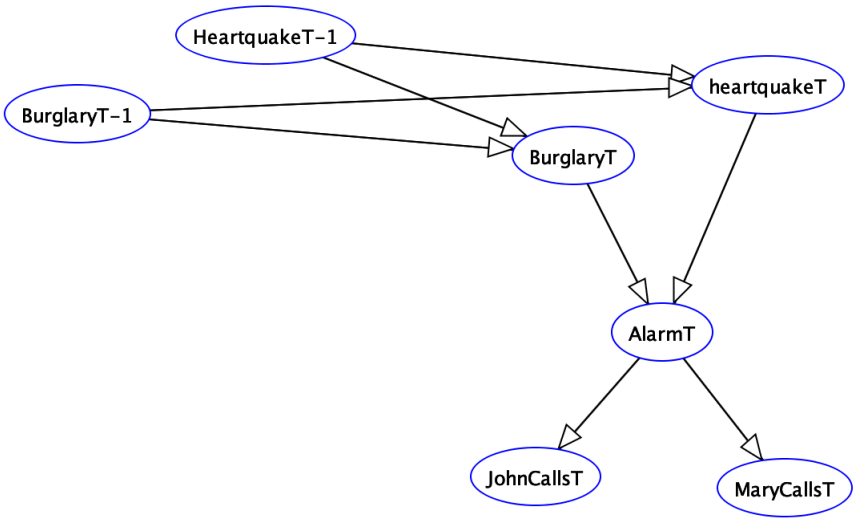


Figure 10: Rappresentzione di uno slice nella DBN Alarm

Infine si è deciso di dinamizzare anche la rete Asia, rendendo come variabili di transizione Asia, Smoke, Lung e Tub e come possibili variabili di evidenza Either, Bronc, Xray o Dysp. Si è posta la query su Smoke al tempo t10 con evidenze su Xray e Dysp.

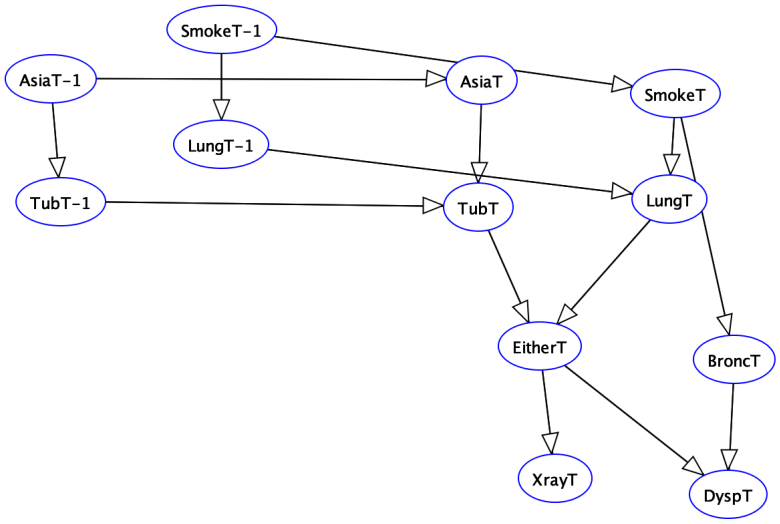
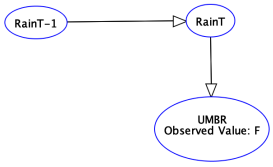


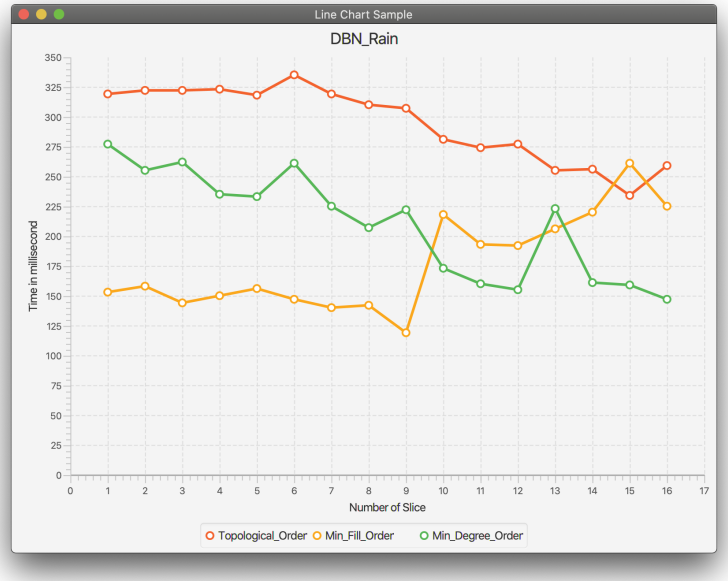
Figure 11: Rappresentazione di uno slice nella DBN Asia

3.3.2 Valutazioni

I seguenti grafici mostrano i risultati ottenuti dalle varie reti con i diversi ordinamenti ponendo come query tutte le variabili di stato: Guardando i grafici, e valutando i tempi medi di esecuzione della VE per slice, diventa subito evidente che all'aumentare del numero di variabili di transizione, e alla loro complessità di relazione il tempo aumenta notevolmente.

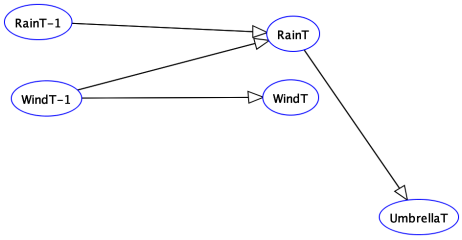
- Rain:
  - Numero nodi di transizione: 1
  - Numero archi di transizione: 1
  - Numero totale di nodi per slice: 3
  - Numero evidenze: 1
  - Numero slice temporali considerati: 16

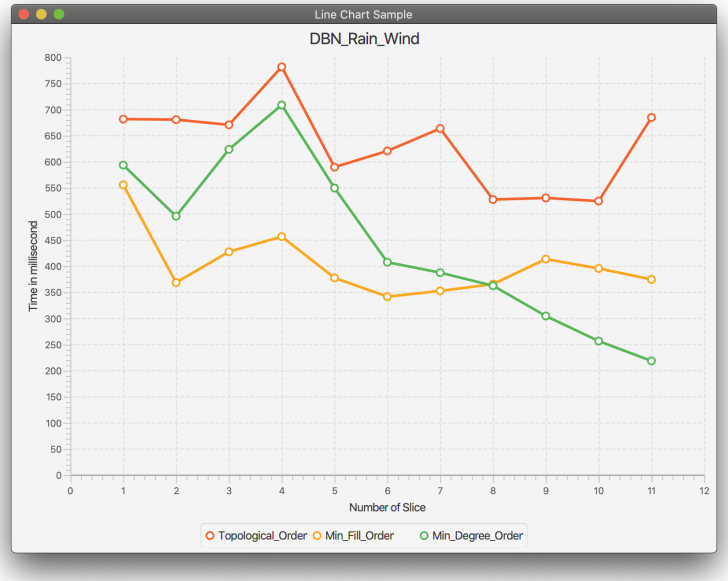




  - è possibile notare come in media il tempo richiesto per la VE è poco meno di 2,5 millisecondi

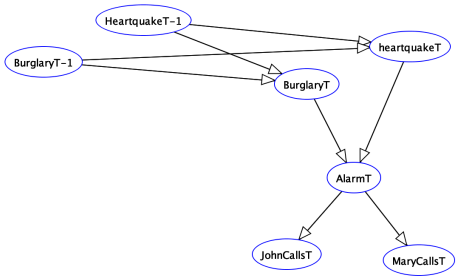
- Wind:
  - Numero nodi di transizione: 1
  - Numero archi di transizione: 1
  - Numero totale di nodi per slice: 3
  - Numero evidenze: 1
  - Numero slice temporali considerati: 10

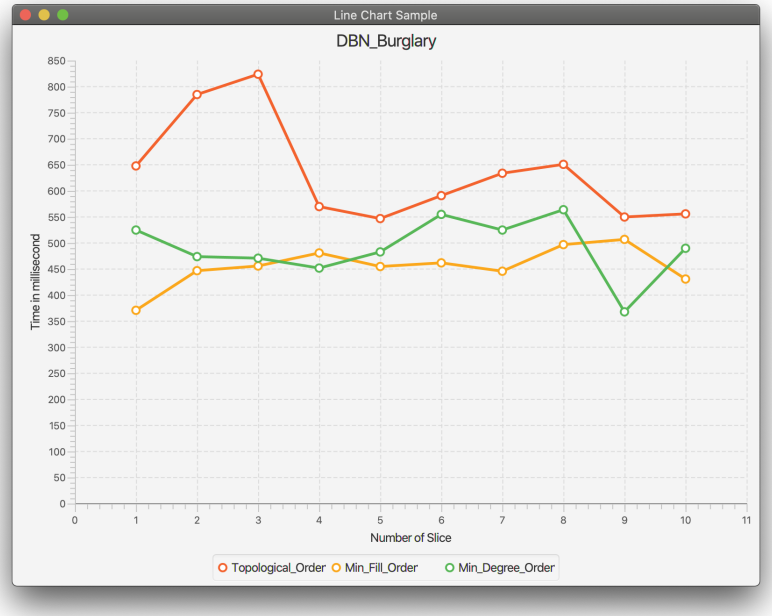




  - essa rappresenta una semplice estensione della precedente con una variabile di stato in più e con una complessità di relazione fra esse maggiore, il tempo sale a circa 6 millisecondi. L'aggiunta di un solo nodo e di due archi di transizione in più ha raddoppiato il tempo di esecuzione.

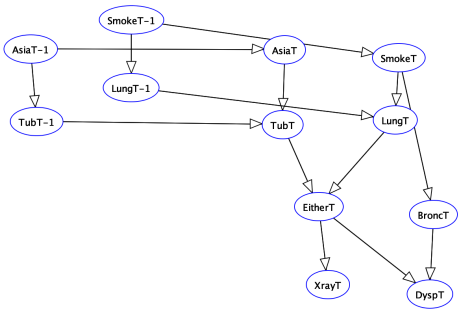
- Alarm:
  - Numero nodi di transizione: 2
  - Numero archi di transizione: 4
  - Numero totale di nodi per slice: 7
  - Numero possibili evidenze: 3
  - Numero slice temporali considerati: 9

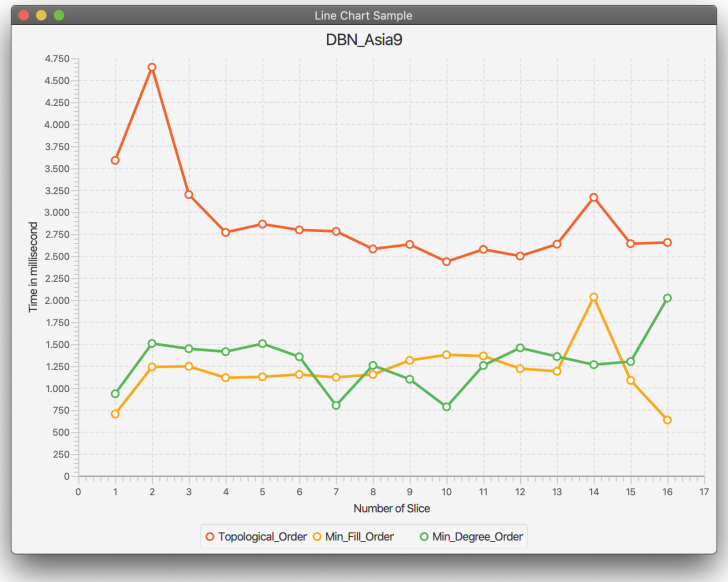




  - Tempo medio di esecuzione: 5.5 millisecondi

- Asia:
  - Numero nodi di transizione: 4
  - Numero archi di transizione: 4
  - Numero totale di nodi per slice: 12
  - Numero possibili evidenze: 4
  - Numero slice temporali considerati: 15





  - Tempo medio di esecuzione: 25 millisecondi. Aumento molto rilevante confronto ai risultati precedenti.

Il grafico in figura 11 ci indica i tempi di inferenza ottenuti sulla rete con complessità maggiore rispetto a quelle valutate, in termini di numero di nodi e di relazione tra stato precedente e stato successivo. Eínfatti possibile notare che il tempo medio di esecuzione della VE è il maggiore di tutti. Potendo considerare l'inferenza a due "slice" di una DBN, l'equivalente della computazione degli stessi nodi considerati in una VE su BN statiche allora quanto detto finora rappresenta una ulteriore conferma degli studi condotti sulle BN statiche. Se il quantitativo di nodi e di relazione fra essi è maggiore allora aumenta la treewidth della rete. La complessità temporale sappiamo essere fortemente dipendente dalla treewidth.

Altra conferma agli studi sulla BN li notiamo all'interno dei grafici vedendo le linee colorate; l'ordinamento delle variabili influisce sul tempo di esecuzione della VE, infatti la linea rossa (ordine topologico) risulta sempre superiore alle linee gialla e verde (min-fill e min-degree rispettivamente). L'ordine topologico è spesso più oneroso rispetto alle tecniche di ordinamento quali min-degree e il min-fill. Tale vantaggio lo si nota sia per via della minor complessità della computazione per la VE e sia per la diminuzione della width a cui tali ordinamenti portano.
Ulteriore valutazione effettuata è stata a riguardo le differenze dei tempi di esecuzione correlata al numero massimo dei fattori intermedi. Si è scelta la rete Alarm, in quanto essa si presta bene al caso da dimostrare, inoltre si è scelto l'ordinamento con euristica `MinFillOrder`.

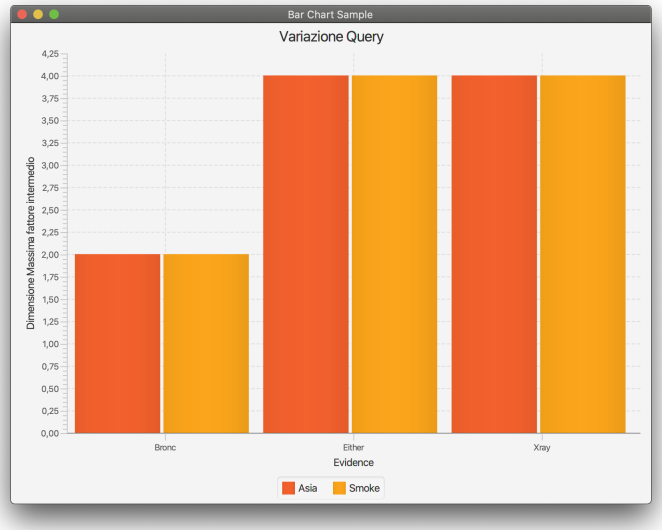


Figure 12: numero massimo fattori intermedi con DBN Asia con diverse query e diverse evidenze

Infatti in tale rete, se scegliessimo come variabile di query Asia o Smoke, dando come evidenza Bronc, avremo che, eliminando i nodi non rilevanti, il risultato della VE ad ogni passaggio è un insieme di fattori in cui al massimo compaiono a coppie Smoke e Lung, e Asia e Tub, quindi con al massimo un fattore contenente 2 variabili. Mentre nel caso di query con evidenza Either o Xray, le 4 variabili di stato sono in relazione tra di loro e infatti avremo un fattore intermedio che rappresenta la distribuzione di probabilità condizionale delle 4 variabili, quindi con un fattore ad ongi slice di dimensione pari a 4. Per questo la dimensione massima del fattore nel grafico in figura 12, rappresenta la width corrispondente alle diverse query e sappiamo che la complessità della variable elimination, nel caso di reti multy-connected, è esponenziale su tale numero e lineare sul numero di nodi $O(d^w n)$.
Infatti avremo che, essendo in un contesto booleano:

- nel primo caso la complessità sarà pari a = $O(2^2 9)$, dove il numero di nodi corrisponde al numero totale di nodi della rete senza i nodi Either, Xray e Dysp che risultano irrilevanti;
- nel secondo caso invece essa sarà pari a $O(2^4 9)$, dove il numero di nodi corrisponde al numero totale di nodi della rete senza i nodi Bronc, Xray e Dysp che risultano irrilevanti;
- nel terzo caso invece essa sarà pari a $O(2^4 10)$, dove il numero di nodi corrisponde al numero totale di nodi della rete senza i nodi Bronc e Dysp che risultano irrilevanti;

Nel grafico in figura 13 è possibile verificare la differenza di complessità appena spiegata;

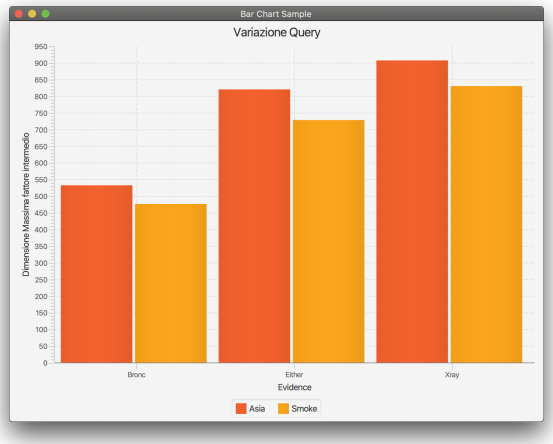


Figure 13: tempo di esecuzione DBN Asia con diverse query e diverse evidenze

Sono state effettuate delle valutazioni di correttezza delle probabilità a posteriori ottenute andando a paragonarle con l'algoritmo di Particle Filtering per l'inferenza approssimata delle DBN. Alcuni esempi:

- Ponendo la query Rain alla rete 3.3.2 al tempo 20 con evidenze scelte randomicamente:

	Risultati ottenuti	
Tecnica di inferenza	true	false
Variable Elimination	0.23	0.77
Particle Filtering	0.14	0.86

Table 1: Inferenze per la query P(RainT20$\|E$)

- Ponendo la query Rain e Wind per la rete in figura 3.3.2 al tempo 10 con evidenze scelte randomicamente:

	Risultati ottenuti	
Tecnica di inferenza	truetrue	falsetrue
Variable Elimination	0.35	0.55
Particle Filtering	0.37	0.53

Table 2: Inferenze per la query P(RainT10,WindT10$\|E$)

- Ponendo la query Burglary e Earthquake per la rete in figura 3.3.2 al tempo 9 con evidenze scelte randomicamente:

	Risultati ottenuti	
Tecnica di inferenza	falsetrue	falsefalse
Variable Elimination	0.18	0.82
Particle Filtering	0.1	0.9

Table 3: Inferenze per la query P(EarthquakeT9,BurglaryT9$\|E$)