Relazione progetto Uncertainty

Andreata Chiara, Gerbaudo Luca, Racca Andrea Rocco

a.a. 2019 - 2020

1 Introduzione

Per poter sviluppare il progetto in maniera ottimale ed eseguire i test necessari è stata fondamentale una parte iniziale di comprensione della struttura e del funzionamento delle reti bayesiane.
Una volta apprese le nozioni abbiamo iniziato a ragionare sulle richieste fatte e su ciò che comportavano sulle reti stesse.
Abbiamo quindi deciso di procedere con ordine e sviluppare inizialmente i metodi di pruning (di archi e nodi), per poi passare agli ordini diversi da quello di default (MinDegreeOrder e MinFillOrder).
Infine, abbiamo concentrato i nostri sforzi sullo sviluppo del rollup filtering sulle reti bayesiane dinamiche per ovviare ad un problema comune denominato "unrolling", espresso dettagliatamente nelle sezioni successive.
Per ciascuna tipologia di esercizio richiesto abbiamo eseguito molti test con reti differenti e con insiemi di variabili di evidenza più o meno grandi: questo per poter apprendere al meglio cosa comportassero le modifiche fatte alla variable elimination (VE) con i pruning e con i vari ordinamenti.

2 RETI BAYESIANE STATICHE

In questa sezione vengono analizzate le operazioni eseguite su reti bayesiane statiche e sui test eseguiti su di esse. Lo scopo è dimostrare il miglioramento dei tempi di esecuzione e l'efficienza degli algoritmi per poter ottenere i risultati desiderati a partire da determinate query.

2.1 Pruning nodi

Il pruning dei nodi è stato realizzando secondo due metodi: quello basato sugli antenati e quello che sfrutta il moral graph.

2.1.1 Pruning basato sugli antenati

Questo metodo dice che una variabile di evidenza Y è irrilevante, tranne nel caso in cui $Y \in Ancestors(\{X\} \cup E)$, dove $Ancestors$ è l'insieme degli antenati della variabile di query X e delle variabili di evidenza presente nell'insieme E.
Dalla definizione, si può dedurre che se le variabili X ed Y si trovano in basso nella rete bayesiana a cui appartengono hanno un grosso numero di antenati e, quindi, sarà più difficile che Y risulti irrilevante poichè più facilmete rientrerà nell'insieme degli $Ancerstors$.
Al contrario, invece, se X ed Y sono in alto nella rete sarà più semplice per Y risultare irrilevante, in quanto le due variabili hanno pochi antenati.

2.1.2 Pruning basato sul moral graph

Con questo metodo, invece, si va a sfruttare il moral graph di una rete bayesiana: grafo che si occupa di "far sposare" tutti i genitori andando a tracciare un arco tra di essi e creando un grafo non diretto.
A questa definizione va aggiunta quella della m-separation, dove: A è m-separata da B tramite C se e solo se sono separate da C nel moral-graph. Da queste due definizioni deriva il teorema che recita: Y è irrilevante se è m-separata da X(variabile di query) attraverso E(evidenza).
Con questo metodo di pruning è quindi facile trovare molte variabili irrilevanti nel caso in cui vi siano molte variabili di evidenza e poche di query. Al contrario, invece, se abbiamo allo stesso tempo un numero esiguo di variabili di evidenza ed un numero elevato di variabili di query, risulterà più difficile collegare Y ad X senza che vi si frapponga una variabile di evidenza.
Come ultima considerazione abbiamo notato che è più semplice trovare nodi irrilevanti con una rete poco connessa, ovvero con un numero ridotto di archi, in cui le variabili di evidenza siano posizionate in modo sparso all'interno della rete bayesiana.

2.2 Pruning archi

L'ultimo metodo proposto pereseguire operazioni di pruning sulle reti bayesiane statiche è stato quello di trovare e potare gli archi irrilevanti.

A partire dalla definizione: per ogni arco $U \rightarrow E$ originato da un nodo U in E si possono eseguire due operazioni:

1. rimuovere l'arco $U \rightarrow E$ dalla rete
2. Rimpiazzare la CPT con una sua versione ridotta che tiene conto del valore u del genitore U dato nell'evidenza e

Con queste operazioni si è in grado di ottenere una rete in grado di computare una query in maniera corretta e senza differenze nel risultato pur trattandosi di una rete semplificata.
Com'è facilmente intuibile, questo metodo di pruning funziona al meglio quando le variabili di evidenza sono numerose, poichè possono esserci più archi irrilevanti ed è possibile andare a ridurre il numero di CPT nel caso in cui le variabili di evidenza abbiano tanti figli.

2.3 Test eseguiti sugli algoritmi di pruning

Per condurre i test per verificare l'efficienza degli algoritmi di pruning realizzati abbiamo preso in considerazione diverse reti bayesiane con un numero più o meno elevato di nodi. Abbiamo poi modificato le variabili di evidenza per vedere le differenze nei risultati ottenuti al termine della computazione dei vari algoritmi realizzati.

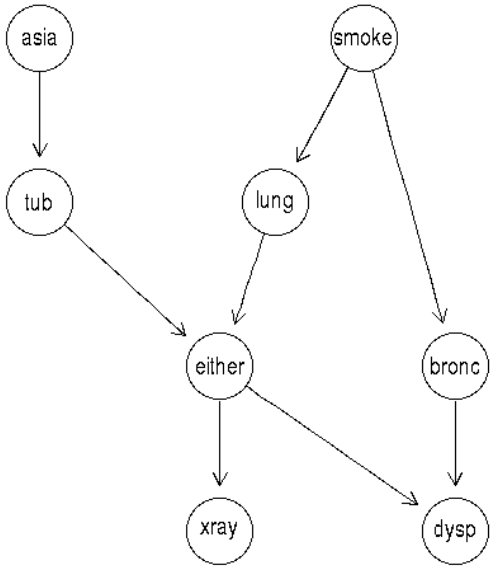

Figure 1: Rete Asia utilizzata per i test

Rete: Asia $\rightarrow P(Asia|Either = "Yes")$

La rete in questione, presa dal sito indicatoci nelle consegne del progetto, è molto basica e presenta solamente 8 nodi. Il primo test è stato effettuato a partire dalla seguente richiesta: $P(Asia|Either = "Yes")$.

- **Metodo di pruning basato sugli antenati:** si ottengono come nodi irrilevanti *Bronch, Dysp e Xray*, confermandoci che l'algoritmo funziona e agisce come ci si aspetta. I tre nodi restituiti come irrilevanti non appartengono infatti all'insieme degli antenati di $Asia$ e *Either*.
- **Metodo di pruning basato sul moral graph:** la computazione porta ad ottenere come nodo irrilevante solamente *Xray*, poichè *Tub e Lung* sono state "sposate" nel moral graph.
- **Metodo di pruning degli archi:** questo algoritmo va a rimuovere 2 archi che risultano essere irrilevanti.

Metodo di pruning	#Irrilevanze	Tempo di esecuzione (ms)
Niente	/	47
Nodi irrilevanti	3	34
M-Separation	1	25
Archi irrilevanti	2	30

Table 1: $P(Asia|Either = "Yes")$ con V.E. Russel & Norvig e ordine topologico inverso

Rete: Asia $\rightarrow P(Smoke|Either = "Yes", Tub = "Yes")$

In questo test abbiamo aumentato il numero di variabili di evidenza, ottendeno risultati leggermente diversi. In particolare, è stato trovato un nodo irrilevante in più con il metodo basato sul moral graph.
Nello specifico:

- **Metodo di pruning basato sugli antenati:** si ottengono come nodi irrilevanti *Bronch, Dysp e Xray*, poichè non fanno parte degli antenati di *Smoke, Either o Tub*.
- **Metodo di pruning basato sul moral graph:** la computazione porta ad ottenere come nodi irrilevanti *Asia ed XRay*.
- **Metodo di pruning degli archi:** questo algoritmo va a rimuovere 3 archi risultati irrilevanti.

Metodo di pruning	#Irrilevanze	Tempo di esecuzione (ms)
Niente	/	29
Nodi irrilevanti	3	23
M-Separation	2	19
Archi irrilevanti	3	22

Table 2: $P(Smoke|Either = "Yes", Tub = "Yes")$ con V.E. Russel & Norvig e ordine topologico inverso

Figure 2: Rete Insurance utilizzata per i test

Rete: Insurance $\rightarrow P(Age|ThisCarDam = "Mild", MedCost = "Thousand", ILiCost = "Thousand", OtherCarCost = "Thousand", ThisCarCost = "Thousand", Theft = "False")$

Abbiamo eseguito test sulla rete Insurace per testare i vari algoritmi di pruning su una rete di dimensioni maggiori.
Come anticipato in precedenza, ponendo come variabili di evidenza i nodi che si posizionano in basso nella rete bayesiana, pur trattandosi di una rete con tanti nodi (27 per la precisione), vengono decretate irrilevanti poche variabili grazie alla presenza di un elevato numero di antenati.

Questi sono stati i risultati ottenuti a partire da
$P(Age|ThisCarDam = "Mild", MedCost = "Thousand", ILiCost = "Thousand", OtherCarCost = "Thousand", ThisCarCost = "Thousand", Theft = "False")$:

- **Metodo di pruning basato sugli antenati:** si ottengono come nodi irrilevanti *OtherCar, GoodStudent, DrivHist, PropCost.*
- **Metodo di pruning basato sul moral graph:** la computazione porta ad ottenere come nodo irrilevante *PropCost.*
- **Metodo di pruning degli archi:** questo algoritmo va a rimuovere 4 archi che risultano irrilevanti.

Metodo di pruning	#Irrilevanze	Tempo di esecuzione (ms)
Niente	/	944
Nodi irrilevanti	4	543
M-Separation	1	594
Archi irrilevanti	4	629

Table 3: $P(Age|ThisCarDam = "Mild", MedCost = "Thousand", ILiCost = "Thousand", OtherCarCost = "Thousand", ThisCarCost = "Thousand", Theft = "False")$ con V.E. Russel & Norvig e ordine topologico inverso

Rete: Insurance $\rightarrow P(Age|SocioEcon = "Prole", GoodStudent = "True")$

Andando invece a selezionare poche variabili di evidenza posizionate in basso nella rete, è possibile notare un importante aumento del numero di nodi irrilevanti trovati con il primo metodo di pruning.
É inoltre possibile notare una drastica riduzione dei tempi di esecuzione che, andando a non considerare 24 nodi, scende da circa 3 secondi a pochi millisecondi.
Di seguito sono mostrati i risultati ottenuti:

- **Metodo di pruning basato sugli antenati:** si ottengono come nodi irrilevanti *Mileage,RiskAversion,OtherCar,SeniorTrain,HomeBase, VehicleYear,MakeModel,AntiTheft,DrivingSkill,RuggedAuto,Airbag, Antilock,CarValue,DrivQuality,DrivHist,Cushioning,Theft,Accident, MedCost,ThisCarDam,OtherCarCost,ILiCost,ThisCarCost,PropCost.*
- **Metodo di pruning basato sul moral graph:** la computazione porta ad ottenere come nodo irrilevante *OtherCar.*
- **Metodo di pruning degli archi:** questo algoritmo va a rimuovere 7 archi risultati irrilevanti.

Metodo di pruning	#Irrilevanze	Tempo di esecuzione (ms)
Niente	/	3644
Nodi irrilevanti	24	16
M-Separation	1	3230
Archi irrilevanti	7	3621

Table 4: $P(Age|SocioEcon = "Prole", GoodStudent = "True")$ con V.E. Russel & Norvig e ordine topologico inverso

Rete: Insurance $\rightarrow P(ThisCarCost|Accident = "None", RuggedAuto = "Football", HomeBase = "Secure", CarValue = "FiveThou", AntiTheft = "False")$

Per eseguire questo test abbiamo provato a selezionare dei nodi in modo strategico per fare in modo che l'algoritmo di pruning basato sul moral graph riuscisse a eliminare più nodi.
I risultati ottenuti confermano le nostre intuizioni in quanto sia con il metodo di pruning basato sugli antenati sia il metodo moral graph vengono potanti molti nodi. Si può però notare che rispetto al test precedente, essendo le variabili di evidenza posizionate relativamente in basso, il metodo di pruning basato sui nodi irrilevanti non è molto prestante.
Di seguito sono mostrati i risultati ottenuti:

- **Metodo di pruning basato sugli antenati:** si ottengono come nodi irrilevanti *Mileage,RiskAversion,OtherCar,SeniorTrain,HomeBase,*

OtherCar,GoodStudent,Airbag,DrivHist,Cushioning,MedCost, OtherCarCost,ILiCost,PropCost.

- **Metodo di pruning basato sul moral graph:** la computazione porta ad ottenere come nodi irrilevanti. *GoodStudent,MedCost,Mileage, RiskAversion,VehicleYear,Cushioning,SeniorTrain,DrivingSkill,Antilock, ILiCost,OtherCar,SocioEcon,MakeModel,Airbag,DrivHist,DrivQuality,Age.*
- **Metodo di pruning degli archi:** questo algoritmo va a rimuovere 11 archi risultati irrilevanti.

Metodo di pruning	#Irrilevanze	Tempo di esecuzione (ms)
Niente	/	335
Nodi irrilevanti	9	213
M-Separation	17	49
Archi irrilevanti	11	307

Table 5: $P(ThisCarCost|Accident = "None", RuggedAuto = "Football", HomeBase = "Secure", CarValue = "FiveThou", AntiTheft = "False")$ con V.E. Russel & Norvig e ordine topologico inverso

2.4 Ordinamenti

Per poter effettuare dei test su diversi ordinamenti delle variabili abbiamo dovuto modificare l'algoritmo Variable Elimination di Darwiche che consente di passare come parametro un ordinamento delle variabili su cui andare ad effettuare il calcolo. Avendo una complessità lineare nel numero di variabili, ma esponenziale nella treewidth, è possibile e utile andare a sfruttare un corretto ordinamento per ridurre il più possibile il fattore più grande che viene generato durante l'algoritmo.

2.4.1 MinDegreeOrder

Il MinDegreeOrder propone un ordinamento basato sull'interaction graph e sul dare la precedenza all'eliminazione della variabile che in tale grafo presenta il minor numero di vicini: questo perchè è probabile che tale variabile vada a contribuire alla costruzione del fattore più piccolo.

2.4.2 MinFillOrder

Quest'altro ordinamento, invece, considera per l'eliminazione la variabile che andrebbe ad aggiungere il minor numero di archi tra i suoi vicini nell'algoritmo di ordinamento precedentemente descritto.
É possibile, ad esempio, che giungendo ad una variabile, la quale nell'interaction graph ha tutti i vicini collegati da archi, si siano già precedentemente esaminati i fattori di tutti i vicini e quindi si potrà evitare di andare a costruire un fattore più grande.

2.5 Test eseguiti sugli algoritmi di ordinamento

Per eseguire i test con i vari ordinamenti delle variabili, abbiamo sfruttato le reti precedentemente analizzate per andare a sottolineare la differenza dei tempi di computazione in base alla rete che si sta analizzando e alle variabili di evidenza utilizzate.

Rete: Asia $\rightarrow P(Asia|Either = "Yes")$

Come ci aspettavamo in questo caso, avendo una rete molto piccola, con un numero ridotto di nodi la $Width$ non cambia e anche il tempo di esecuzione non varia molto, nonostante salti subito all'occhio che il tempo di esecuzione di MinDegree è il più alto.

Ordinamento	Width	Tempo di esecuzione (ms)
Topologico Inverso (R&S)	2	47
Topologico Inverso (Drw)	2	5
MinDegree	2	51
MinFill	2	10

Table 6: $P(Asia|Either = "Yes")$ senza pruning

Rete: Asia $\rightarrow P(Smoke|Either = "Yes", Tub = "Yes")$

Come per il test precedente, trattandosi di una rete piccola e semplice, non vi sono differenze nella $Width$. Anche in questo caso, $MinDegree$ porta ad un tempo di esecuzione superiore agli altri.

Ordinamento	Width	Tempo di esecuzione (ms)
Topologico Inverso (R&S)	2	29
Topologico Inverso (Drw)	2	5
MinDegree	2	32
MinFill	2	8

Table 7: $P(Smoke|Either = "Yes", Tub = "Yes")$ senza pruning

Rete: Insurance $\rightarrow P(Age|ThisCarDam = "Mild", MedCost = "Thousand", ILiCost = "Thousand", OtherCarCost = "Thousand", ThisCarCost = "Thousand", Theft = "False")$

Sfruttando la rete insurance utilizzata per i test sui metodi di pruning, abbiamo eseguito delle prove anche con i nuovi algoritmi di ordinamento realizzati, andando a notare dei risultati interessanti.
In questo test abbiamo utilizzato la stessa richiesta che presenta una sola variabile di query e numerose variabili di evidenza. In questo caso è possibile notare come le $Width$ varino, così come i tempi di esecuzione. Risulta evidente

l'effettiva efficienza degli ordinamenti delle variabili.
I dettagli sono elencati nella tabella sottostante:

Ordinamento	Width	Tempo di esecuzione (ms)
Topologico Inverso (R&S)	4	563
Topologico Inverso (Drw)	8	173
MinDegree	6	98
MinFill	6	75

Table 8: $P(Age|ThisCarDam = "Mild", MedCost = "Thousand", ILiCost = "Thousand", OtherCarCost = "Thousand", ThisCarCost = "Thousand", Theft = "False")$ senza pruning

Rete: Insurance $\rightarrow P(Age|SocioEcon = "Prole", GoodStudent = "True")$

Andando a inserire poche variabili di evidenza, l'efficienza nelle tempistiche salta ancora più all'occhio, sottolineando l'effettiva utilità nell'utilizzare un ordinamento diverso da quello topologico inverso.

Ordinamento	Width	Tempo di esecuzione (ms)
Topologico Inverso (R&S)	4	3542
Topologico Inverso (Drw)	11	2377
MinDegree	6	72
MinFill	6	53

Table 9: $P(Age|SocioEcon = "Prole", GoodStudent = "True")$ senza pruning

I seguenti test sono stati effettuati applicando prima algoritmo di pruning sui nodi irrilevanti e in seguito i vari algoritmi di ordinamento.
Generalmente eseguire prima operazioni di pruning rende meno rilevante l'effetto degli algoritmi di ordinamento. Infatti la $Width$ risulta la stessa per ciascun algoritmo di ordinamento applicato, mentre per quanto riguarda il tempo di esecuzione di MinDegree order si ha un rallentamento dell'esecuzione.
I risultati ottenuti sono stati paragonati a quelli già ottenuti testando la rete Insurance (pag 7, tabella 4).

Ordinamento	Width	Tempo di esecuzione (ms)
Topologico Inverso (R&S)	1	18
Topologico Inverso (Drw)	1	1
MinDegree	1	103
MinFill	1	3

Table 10: $P(Age|SocioEcon = "Prole", GoodStudent = "True")$ con pruning nodi irrilevanti (#24)

Abbiamo provato anche a comparare l'esecuzione degli ordinamenti dopo aver effettuato il pruning basato su M-Separation. In questo caso, siccome viene trovato un solo nodo irrilevante, le prestazioni non hanno un evidente miglioramento rispetto al caso senza pruning.

Ordinamento	Width	Tempo di esecuzione (ms)
Topologico Inverso (R&S)	4	3538
Topologico Inverso (Drw)	11	2419
MinDegree	6	41
MinFill	6	38

Table 11: $P(Age|SocioEcon = "Prole", GoodStudent = "True")$ con pruning M-Separation (#1)

Per una maggior completezza, abbiamo effettuato anche un test sugli ordinamenti dopo aver effettuato il pruning sugli archi irrilevanti e, come per M-Separation, non vi sono evidenti miglioramenti dei tempi di esecuzione.

Ordinamento	Width	Tempo di esecuzione (ms)
Topologico Inverso (R&S)	4	3473
Topologico Inverso (Drw)	11	2298
MinDegree	7	79
MinFill	6	59

Table 12: $P(Age|SocioEcon = "Prole", GoodStudent = "True")$ con pruning Archi irrilevanti (#7)

Rete: Insurance $\rightarrow P(ThisCarCost|Accident = "None", RuggedAuto = "Football", HomeBase = "Secure", CarValue = "FiveThou", AntiTheft = "False")$

Anche in questo caso, avendo molte variabili di evidenza, cambiando il tipo di ordinamento, le tempistiche non variano molto; solo nel caso di MinFill order i tempi sono drasticamente ridotti. Combinando il pruning basato sul Moral

Ordinamento	Width	Tempo di esecuzione (ms)
Topologico Inverso (R&S)	3	260
Topologico Inverso (Drw)	6	164
MinDegree	4	239
MinFill	4	27

Table 13: $P(ThisCarCost|Accident = "None", RuggedAuto = "Football", HomeBase = "Secure", CarValue = "FiveThou", AntiTheft = "False")$ senza pruning

Graph e gli algoritmi di ordinamento, le prestazioni non variano molto in quanto

molti nodi vengono potati, anzi con il MinFill order che prima risultava il più performante ora risulta il più lento.

Ordinamento	Width	Tempo di esecuzione (ms)
Topologico Inverso (R&S)	3	17
Topologico Inverso (Drw)	4	35
MinDegree	4	36
MinFill	4	52

Table 14: $P(ThisCarCost|Accident = "None", RuggedAuto = "Football", HomeBase = "Secure", CarValue = "FiveThou", AntiTheft = "False")$ con M-Separation (#17)

2.6 Very Large Network

Come ultimi test sulle reti bayesiane abbiamo deciso di prendere due reti con un elevato numero di nodi e verificare che impatto effettivo avessero gli algoritmi di pruning e ordinamento su un grosso di variabili.

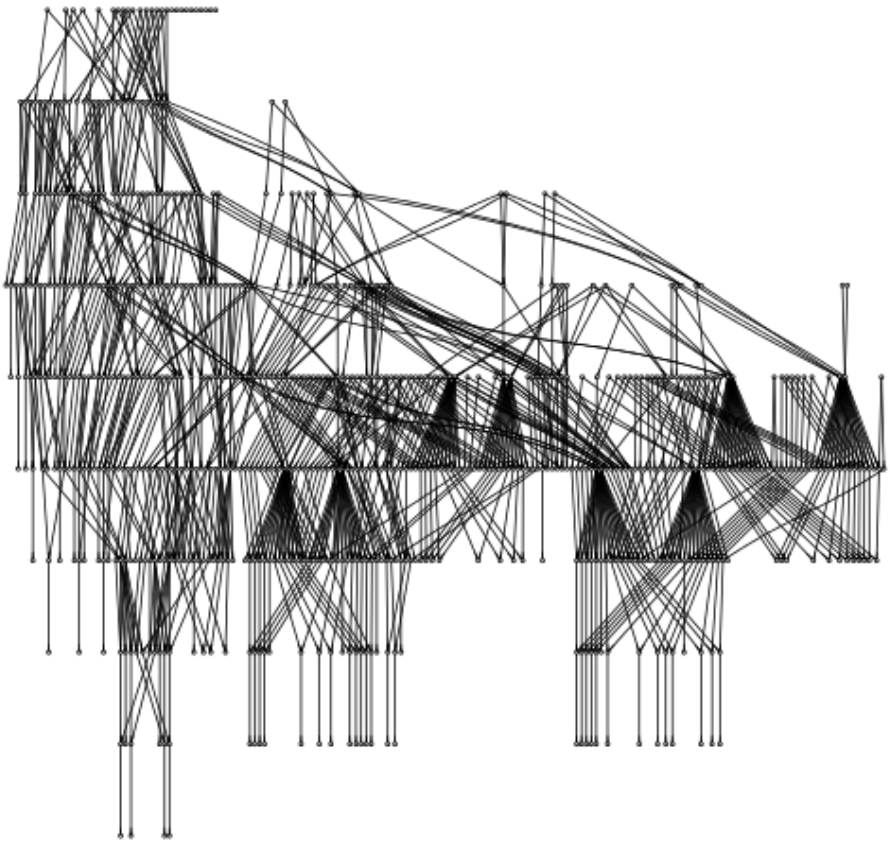

Figure 3: Rete Link utilizzata per i test, 924 nodi

Rete: Link $\rightarrow P(N26\_a\_f | Z\_42\_a\_f = "F", D0\_58\_a\_x = "X")$

Utilizzando questa rete da 924 nodi, eravamo consci del fatto che sarebbero sorti non pochi problemi.
Infatti, senza utilizzare i metodi di pruning, i tempi computazionali sono molto elevati e Java restituisce l'errore "*Java heap space*".
Invece, utilizzando gli algoritmi di pruning e applicando entrambi gli ordianmenti realizzati (MinFill e MinDegree), la computazione termina in pochi *ms*,

rispettivamente (circa) 753 e 457, in quanto l'algoritmo di pruning dei nodi basato sugli antenati individua ben 620 nodi irrilevanti.

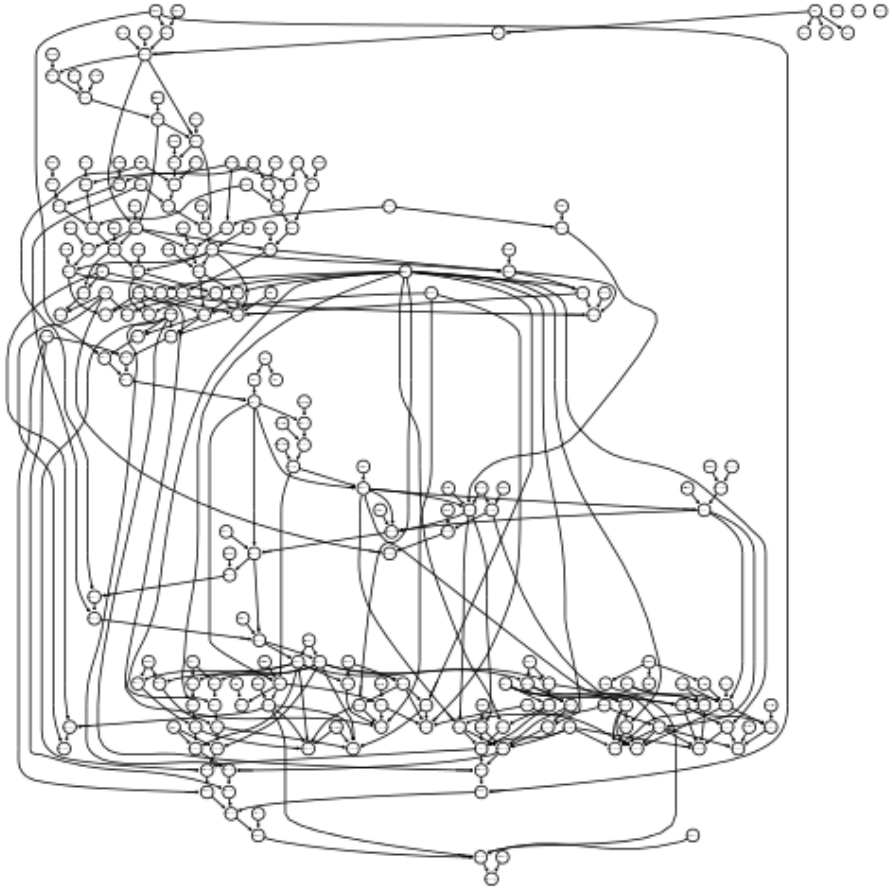

Figure 4: Rete Andes utilizzata per i test, 223 nodi

Rete: Andes $\rightarrow P(NORMAL52|TRY76 = "False", GOAL\_56 = "False", SNode\_24 = "False")$

Per analizzare al meglio il problema delle reti bayesiane di grande dimensione, abbiamo deciso di testare una rete più piccola (figura 4) rispetto a quella vista in precedenza (figura 3) per cogliere altri particolari significativi.

Testando questa rete senza algoritmi di pruning e con ordinamento Topologico Inverso (R&S) e Darwiche, dopo più di 10 minuti di computazione abbiamo ragionevolmente interrotto l'esecuzione.
Invece, utilizzando gli algoritmi di ordinamento MinDegree e MinFill, l'esecuzione è terminata correttamente dopo pochi secondi, più precisamente dopo (circa) $20982ms$ e $3891ms$.
Abbiamo quindi constatato che, nel caso in cui si lavori con reti di elevate dimensioni, gli algoritmi di pruning risultano molto utili in quanto vanno ad eliminare molti nodi (219 in questo caso), rendendo l'esecuzione decisamente rapida con tutti gli algoritmi di ordinamento analizzati.
Tali deduzioni sono ben visibili nella tabella sottostante:

Ordinamento	Width	Tempo di esecuzione (ms)
Topologico Inverso (R&S)	5	309
Topologico Inverso (Drw)	10	87
MinDegree	5	23
MinFill	5	15

Table 15: $P(ThisCarCost|Accident = "None", RuggedAuto = "Football", HomeBase = "Secure", CarValue = "FiveThou", AntiTheft = "False")$ con M-Separation (#17)

2.7 PolyTree

Un polytree è un grafo aciclico diretto che può essere rappresentato con un albero, in altre parole è un grafo con al massimo un percorso non orientato tra due nodi e quindi non presenta cicli.

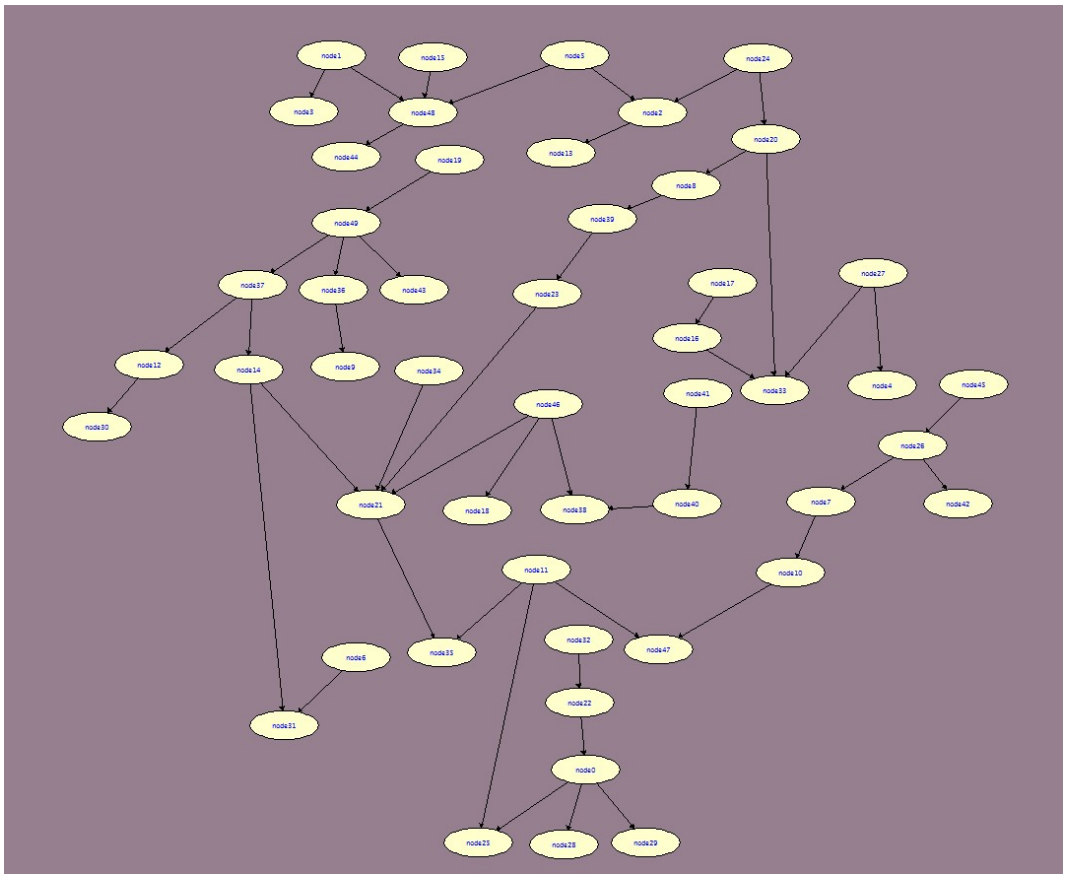

Figure 5: PolyTree utilizzato durante i test.

Rete: MyPolyTree $\rightarrow P(Node47|Node1 = "State0", Node16 = "State0")$
Analisi sui metodi di pruning

Come ci aspettavamo, nel caso del pruning dei nodi irrilevanti sono state rimosse molte variabili (41 per la precisione), mentre con gli algoritmi che sfruttano la M-Separation e gli archi irrilevanti sono stati rimossi pochi elementi (rispettivamente 2 e 3).
Analizzando ulteriormente i dati, si può notare come i tempi di esecuzione non migliorino molto, anzi, per la M-Separation aumentano drasticamente. Questo

conferma le nostre aspettative: i metodi di pruning risultano inefficaci, se non addirittura peggiori, rispetto a prima su un PolyTree.

Metodo di pruning	#Irrilevanze	Tempo di esecuzione (ms)
Niente	/	64
Nodi irrilevanti	41	36
M-Separation	2	96
Archi irrilevanti	3	64

Table 16: $P(Node47|Node1 = "State0", Node16 = "State0")$ on V.E. Russel & Norvig e ordine topologico inverso

Rete: MyPolyTree $\rightarrow P(Node47|Node1 = "State0", Node16 = "State0")$
Analisi sui metodi di ordinamento

Anche in questo caso, come ci aspettavamo, la $Width$ non cambia molto: solo nel caso del topologico inverso è 7, mentre per tutti gli altri è 5.
Analizzando i dati, si può notare che i tempi di esecuzione non migliorano molto, anzi con MinDeegree Order aumentano drasticamente. Questo perchè gli ordinamenti su un PolyTree risultano inefficaci se non addirittura con risultati peggiori rispetto a prima, come spiegato in precedenza.

Ordinamento	Width	Tempo di esecuzione (ms)
Topologico Inverso (R&S)	5	83
Topologico Inverso (Drw)	7	22
MinDegree	5	90
MinFill	5	20

Table 17: $P(Node47|Node1 = "State0", Node16 = "State0")$ senza pruning

Rete: MyPolyTree $\rightarrow P(Node47|Node1 = "State0", Node16 = "State0")$
Analisi sui metodi di pruning combinati con i metodi di ordinamento

Per analizzare in modo più approfondito i metodi di pruning e gli algoritmi di ordinamento su un PolyTree abbiamo deciso di combinare le due cose. I risultati ottenuti sono migliori quando viene combinato il pruning sui nodi irrilevanti e l'ordianmento di Darwiche o il MinFill, mentre con la M-Separation non vi sono evidenti miglioramenti in quanto vengono rimossi solo 2 nodi.

Ordinamento	Width	Tempo di esecuzione (ms)
Topologico Inverso (R&S)	3	37
Topologico Inverso (Drw)	3	3
MinDegree	3	38
MinFill	3	3

Table 18: $P(Node47|Node1 = "State0", Node16 = "State0")$ con pruning nodi irrilevanti (#41)

Ordinamento	Width	Tempo di esecuzione (ms)
Topologico Inverso (R&S)	5	162
Topologico Inverso (Drw)	7	14
MinDegree	5	19
MinFill	5	15

Table 19: $P(Node47|Node1 = "State0", Node16 = "State0")$ con pruning M-Separation (#2)

3 RETI BAYESIANE DINAMICHE

La seconda parte del progetto richiede di lavorare sulle reti bayesiane dinamiche (DBN), caratterizzate da una modellazione temporale in cui, ciascun time-slice, è condizionalmente dipendente dal precedente.
Per poter utilizzare una DBN si ha bisogno di tre elementi in particolare:

1. $P(X_0) \rightarrow$ probabilità a priori sulle variabili di stato
2. $P(X_{t_{+1}}|X_t) \rightarrow$ modello di transizione
3. $P(E_t|X_t) \rightarrow$ modello sensoriale

Come consigliato nelle consegne del progetto, per poter lavorare sulle DBN abbiamo sfruttato alcune reti utilizzate nella prima parte (quindi statiche) e abbiamo proceduto con la loro "dinamizzazione". Per implementare queste reti e gli elementi richiesti abbiamo sfruttato la classe DynamicBayesNet offerta da aima.core. Nonostante fosse ben strutturata, è stato necessario apportare alcune modifiche a tale classe poichè gestisce le transazioni tramite una mappa del tipo $Map < RandomVariable, RandomVariable >$ che però non permetteva ad una variabile di condizionarne più di una al tempo successivo (cosa che invece capita come nell'esempio visto di Wind,Rain ed Umbrella). La modifica principale apportata alla classe, quindi, è stata quella di permettere di sfruttare una lista di variabili contenente tutti gli elementi condizionati al tempo successivo.
Lavorare sulle DBN è possibile tramite l'inferenza che può essere eseguita tramite l'algoritmo "rollup filtering" volto a risolvere il problema di cui soffrono le DBN: l'unrolling. Essendo costruite su un numero illimitato di intervalli, sfruttare la memoria per salvare tutta la rete diventa impensabile e, soprattutto, rappresenterebbe uno spreco di risorse. Per questo motivo, il "rollup filtering" fornisce una soluzione basata sulla focalizzazione su due time-slice alla volta: permette quindi di calcolare i fattori necessari ai tempi t-1 e t per poi proseguire e "dimenticare" i nodi precedenti, che risultano non rilevanti per la computazione dal tempo t in poi.
Come suggerito nelle consegne del progetto, abbiamo modificato l'algoritmo della variable elimination (versione di Darwiche) implementato per la prima parte del nostro lavoro: questo perchè è necessario restituire un insieme di fattori, non il singolo fattore come previsto nella VE. In questo algoritmo, infatti, non si dovranno andare a moltiplicare i fattori per ottenere un unico elemento finale. In particolare, al primo passo di esecuzione l'insieme dei fattori in input all'algoritmo sarà vuoto e, quindi, verrà sfruttata $P(X_0)$, mentre per i passi successivi esso sarà il risultato della computazione ed esecuzione del passo precedente. Questo funzionamento è ben visibile nell'immagine sottostante.

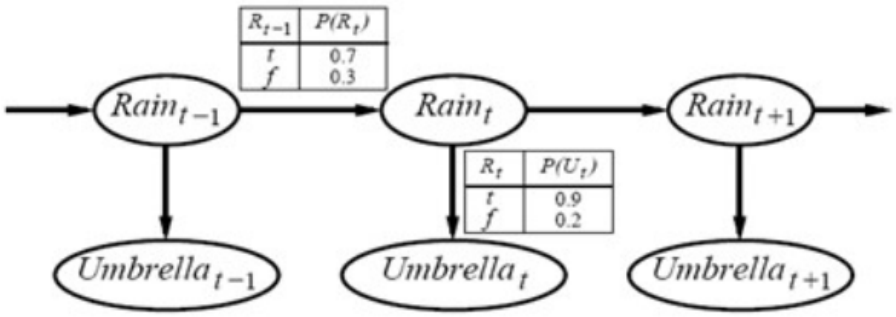

Figure 6: Funzionamento rollup filtering

3.1 Test eseguiti

Per eseguire i test sulle DBN abbiamo sfruttato delle reti fornite dalla libreria aima.core (come quella di Umbrella, Wind e Rain) e abbiamo poi provveduto a "dinamicizzarne" altre, usando soprattutto quelle sfruttate per eseguire i test nella prima parte del progetto.
Il numero di time-slice utilizzato è 5.

Rete: Umbrella, Wind e Rain

Come prima rete di test abbiamo utilizzato quella usata negli esempi durante le lezioni sulle DBN, dove Rain al tempo t è condizionato da Wind e Rain al tempo t-1. Essendo una rete semplice, i tempi di esecuzione per ciascun slice ti tempo sono davvero ridotti (nell'ordine di pochi ms), così com'è piccola la *width* sfruttando ciascun tipo di ordinamento che rimane sempre uguale a 3.

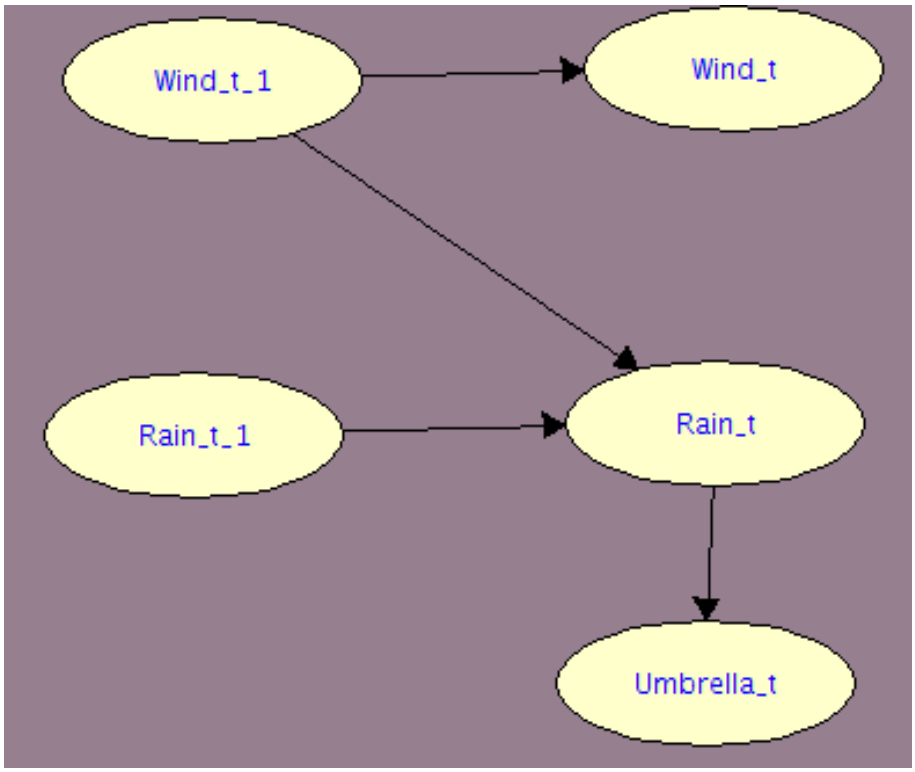

Figure 7: DBN Wind, Rain e Umbrella fornita dalla libreria aima.core

Rete: Alarm dinamicizzata: $P(Burglary, Earthquake, Alarm|JhonCalls = Ture, MaryCalls = true)$

La prima rete presa in considerazione è la versione "dinamicizzata" della semplice rete bayesiana alarm, vista a lezione e usata come esempio frequentemente. Essendo di piccole dimensioni, non sorgono sostanziali differenze provando ad eseguire le query con i diversi algoritmi di pruning e/o ordinamento.
Avendo tempi di esecuzioni quasi irrilevanti (pochissimi ms), i risultati di questi test si basano sul sottolineare la *width* che rimane identica per tutti i tipi di ordinamento, come indicato dal grafico sottostante.

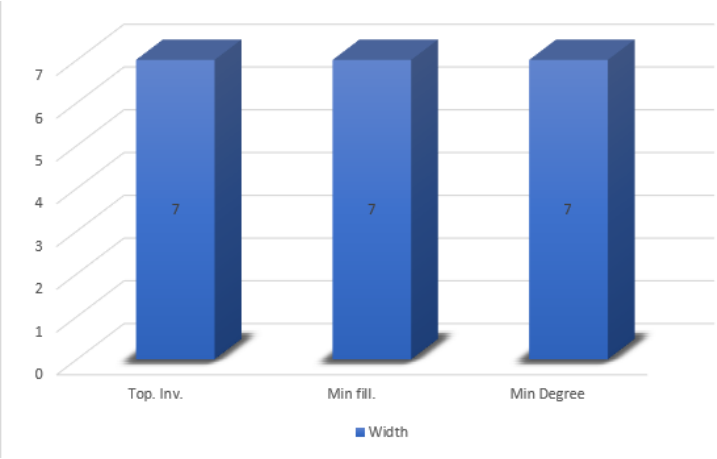

Figure 8: Width per la query: $P(Burglary, Earthquake, Alarm|JhonCalls = Ture, MaryCalls = true)$

Rete: Insurance dinamicizzata: $P(Age|PropCost = Thousand, MedCost = TenThou, ILiCost = HundredThou)$

Con questa rete, invece, è possibile notare differenze le prime differenze: con l'ordine topologico inverso la *width* è nettamente superiore (quasi il doppio) rispetto a quella sorta dagli algoritmi MinFillOrder e MinDegreeOrder che risulta uguale (7).
Inoltre, abbiamo osservato come anche i tempi di esecuzione iniziano a mostrare un grosso divario: con l'ordine topologico inverso, la computazione per eseguire la query in ciascun slice di tempo ha una durata di alcuni secondi (circa 8), mentre con gli ordinamenti da noi realizzati impiega pochi millisecondi. Infine, abbiamo notato come, andando a sfruttare l'algoritmo di pruning sui nodi basato sugli antenati, i tempi di esecuzione calino drasticamente (pur trovando solamente 3 nodi irrilevanti) passando dai 5 secondi di prima a poco più di 1 per quanto riguarda la computazione fatta sulle variabile in ordine topologico inverso.
Sfruttando M-Separation e l'eliminazione degli archi (che non va a cancellarne nessuno), invece, non vi sono ottimizzazioni.

Ordinamento	Width	Tempo di esecuzione (ms)
Topologico Inverso	13	8587
MinDegree	7	78
MinFill	7	85

Table 20: $P(Age|PropCost = Thousand, MedCost = TenThou, ILiCost = HundredThou)$ senza pruning

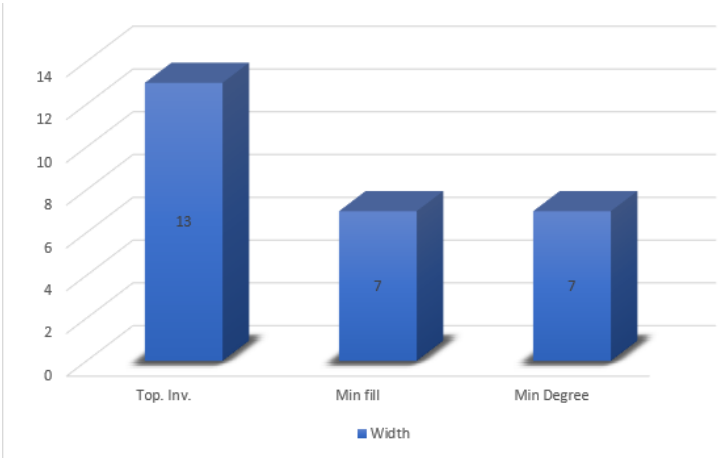

Figure 9: Width per la query: $P(Age|PropCost = Thousand, MedCost = TenThou, ILiCost = HundredThou)$ senza pruning

Ordinamento	Width	Tempo di esecuzione (ms)
Topologico Inverso	12	2257
MinDegree	7	75
MinFill	7	85

Table 21: $P(Age|PropCost = Thousand, MedCost = TenThou, ILiCost = HundredThou)$ con pruning basato sui nodi irrilevanti

Ordinamento	Width	Tempo di esecuzione (ms)
Topologico Inverso	13	7996
MinDegree	7	75
MinFill	7	90

Table 22: $P(Age|PropCost = Thousand, MedCost = TenThou, ILiCost = HundredThou)$ con pruning basato su moral graph

Ordinamento	Width	Tempo di esecuzione (ms)
Topologico Inverso	13	7006
MinDegree	7	80
MinFill	7	90

Table 23: $P(Age|PropCost = Thousand, MedCost = TenThou, ILiCost = HundredThou)$ con pruning basato su archi irrilevanti

Rete: Insurance dinamicizzata con più query variable
$P(Age, MakeModel, CarValue|MedCost = Thousand, ILiCost = TenThou, PropCost = Million)$

Con questa rete, lo stato preso in considerazione comprende le tre query variable: Age, MakeModel e CarValue. Anche in questo caso le tempistiche e la width con

l'ordine topologico inverso (rispettivamente circa 12 secondi per ciascun time-slice e $width = 12$) sono superiori rispetto a quelle ottenute con gli ordinamenti MinFillOrder e MinDegreeOrder. Avendo una query più complessa, i tempi di esecuzione superano gli otto secondi per ciascun time-slice per la computazione eseguita senza lo sfruttamento di ordinamenti diversi da quello di default.
Andando a sfruttare l'algoritmo di pruning basato sugli antenati, nuovamente, si ha un netto miglioramento delle prestazioni, portando i tempi d'esecuzione a calare drasticamente passando a poco più di due secondi per ciascun slice di tempo con l'ordine topologico inverso.

Ordinamento	Width	Tempo di esecuzione (ms)
Topologico Inverso	12	12722
MinDegree	7	88
MinFill	7	90

Table 24: $P(Age, MakeModel, CarValue|MedCost = Thousand, ILiCost = TenThou, PropCost = Million)$ senza pruning

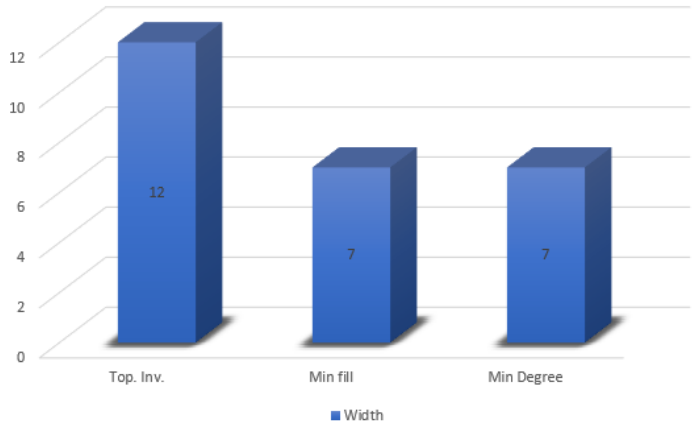

Figure 10: Width per la query: $P(Age, MakeModel, CarValue|MedCost = Thousand, ILiCost = TenThou, PropCost = Million)$ senza pruning

Ordinamento	Width	Tempo di esecuzione (ms)
Topologico Inverso	11	5218
MinDegree	7	86
MinFill	7	89

Table 25: $P(Age, MakeModel, CarValue|MedCost = Thousand, ILiCost = TenThou, PropCost = Million)$ con pruning basato sui nodi irrilevanti

Ordinamento	Width	Tempo di esecuzione (ms)
Topologico Inverso	12	13908
MinDegree	7	92
MinFill	7	111

Table 26: $P(Age, MakeModel, CarValue|MedCost = Thousand, ILiCost = TenThou, PropCost = Million)$ con pruning basato su moral graph

Ordinamento	Width	Tempo di esecuzione (ms)
Topologico Inverso	12	13069
MinDegree	7	98
MinFill	7	83

Table 27: $P(Age, MakeModel, CarValue|MedCost = Thousand, ILiCost = TenThou, PropCost = Million)$ con pruning basato su archi irrilevanti

Rete: Insurance dinamicizzata con più query variable e più evindeze
$P(Age, MakeModel, CarValue|MedCost = Thousand, ILiCost = TenThou, PropCost = Million, Theft = "True", HomeBase = "Secure", OtherCarCost = "Thousand")$

Con la presenza di un numero più elevato di evidenze, i tempi con e senza l'utilizzo degli algoritmi di pruning non cambiano molto, così come per la *width* che cambia di un solo valore tra una test e l'altro.
Questi risultati sono ben visibili nelle tabelle sottostanti

Ordinamento	Width	Tempo di esecuzione (ms)
Topologico Inverso	10	2791
MinDegree	6	37
MinFill	6	32

Table 28: $P(Age, MakeModel, CarValue|MedCost = Thousand, ILiCost = TenThou, PropCost = Million, Theft = "True", HomeBase = "Secure", OtherCarCost = "Thousand")$ senza pruning

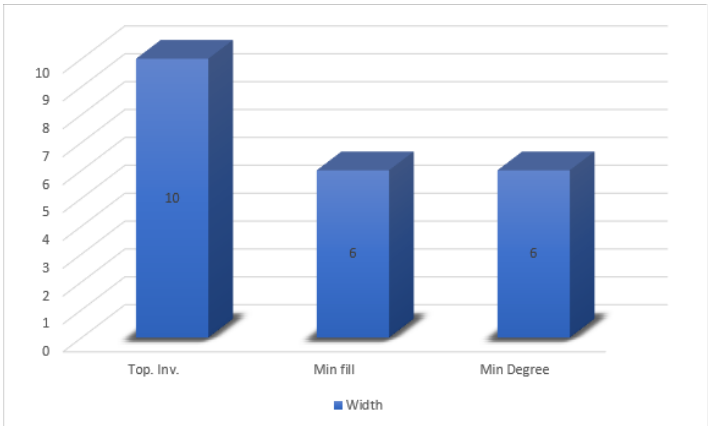

Figure 11: Width per la query: $P(Age, MakeModel, CarValue|MedCost = Thousand, ILiCost = TenThou, PropCost = Million, Theft = \text{"}True\text{"}, HomeBase = \text{"}Secure\text{"}, OtherCarCost = \text{"}Thousand\text{"})$ senza pruning

Ordinamento	Width	Tempo di esecuzione (ms)
Topologico Inverso	9	1155
MinDegree	6	39
MinFill	6	42

Table 29: $P(Age, MakeModel, CarValue|MedCost = Thousand, ILiCost = TenThou, PropCost = Million, Theft = \text{"}True\text{"}, HomeBase = \text{"}Secure\text{"}, OtherCarCost = \text{"}Thousand\text{"})$ con pruning basato sui nodi irrilevanti

Ordinamento	Width	Tempo di esecuzione (ms)
Topologico Inverso	10	2913
MinDegree	6	33
MinFill	6	39

Table 30: $P(Age, MakeModel, CarValue|MedCost = Thousand, ILiCost = TenThou, PropCost = Million, Theft = "True", HomeBase = "Secure", OtherCarCost = "Thousand")$ con pruning basato su moral graph

Ordinamento	Width	Tempo di esecuzione (ms)
Topologico Inverso	10	2702
MinDegree	5	25
MinFill	5	17

Table 31: $P(Age, MakeModel, CarValue|MedCost = Thousand, ILiCost = TenThou, PropCost = Million, Theft = "True", HomeBase = "Secure", OtherCarCost = "Thousand")$ con pruning basato su archi irrilevanti

4 CONCLUSIONI

In conclusione, possiamo affermare che gli algoritmi di pruning riescono a migliorare le prestazioni dell'algoritmo della Variable Elimination nella maggior parte dei casi. In particolar modo si è notato che il primo metodo di pruning dei nodi (quello basato sugli antenati) risulti essere quello più efficace ed efficiente, poichè trova numeri elevati di nodi irrilevanti, in una situazione ottimale, e facendo calare drasticamente i tempi di computazione.
Le differenze si notano meno con reti semplici e, soprattutto, con i polytree dove andare ad eseguire algoritmi di pruning risulta inefficace o addirittura portano a peggiorare i tempi d'esecuzione.

Eseguendo test sulle DBN, analizzando 2 time-slice per volta, invece, abbiamo capito come il rollup filtering permetta di ridurre la complessità spaziale sul numero di nodi della rete: questo è possibile proprio perchè evita di "srotolare" la rete fino al tempo t.