Implementation of Junction Tree Inference Algorithm

COSC 419 Project
Jordan Emslie - 20600152

Introduction

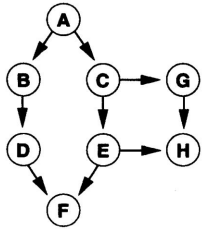
I am interested in this project to implement Junction tree inference algorithm in Typescript. The main reason being, I am interested in the space of inference and prediction so being able to investigate the semantics of how the Junction Tree Inference Algorithm seemed quite neat. I choose to develop in Typescript as I've become fairly proficient with this language after developing with it over the summer. Also, it does not require you to be strictly typed.

Overview

- Create Entities
- Set Entity Dependencies
- Set Entity CPT's
- Build Directed Acyclic Graph (DAG)
- Create Bayesian Network
- Create JTree
    - Graph Transformation (Moral Graph, Triangulated Graph, Cliques, Optimized JTree)
    - Initialization (Inconsistent JTree)
    - Propagation (Consistent JTree)
    - Marginalization

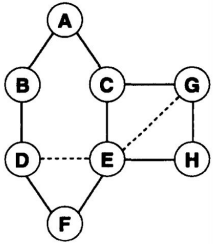
Data Structures

Directed Graph for DAG



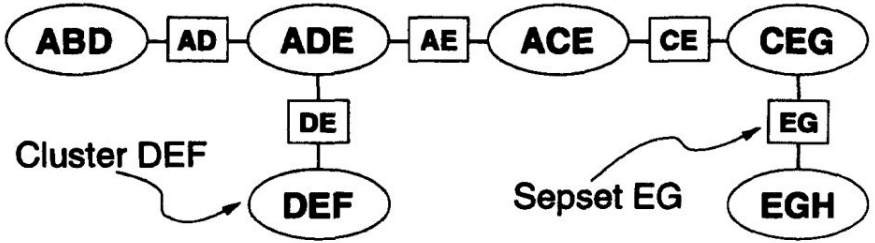
(Huang 234)

UnDirected Graph for Moral Graph



(Huang 234)

Forest-like structure for



(Huang 233)

Focus - JTree's Graphical Transformation Steps

Goal of this functionality is to build the Optimized Junction Tree

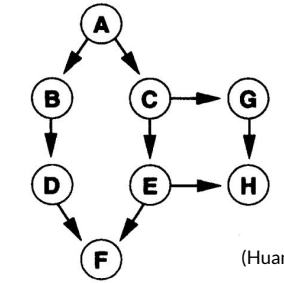
Steps to take:

- Build Moral Graph from DAG
- Build Triangulated Graph and find Cliques from Moral Graph
- Build Optimized Junction Tree with Cliques

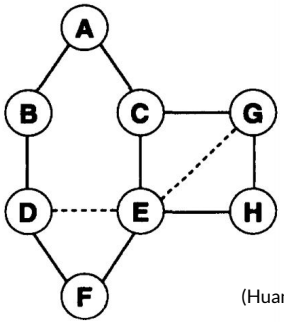
Build Moral Graph from DAG

- Create Undirected version of DAG

- Create Moral Arcs
  - For each entity, get it's parents and if has more than one parent connect parents



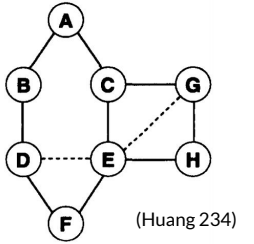
(Huang 234)



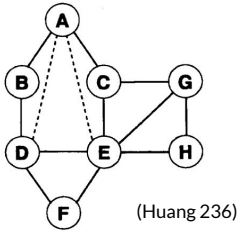
(Huang 234)

Build Triangulated Graph and find Cliques from Moral Graph

- Create a copy of the Moral Graph
- While still entities left in this copy:
  - Select entity:
    - That causes the least number of edges to be added in the next step
    - Tie Breakers will result in picking the entity that induces the cluster with the smallest weight.
- *Weight being:*
  - Weight of entity V is number of values in V
  - Weight of cluster is product of all entities weights in cluster



(Huang 234)



(Huang 236)

Triangulated Graph

Eliminated Vertex	Induced Cluster	Edges Added
H	EGH	none
G	CEG	none
F	DEF	none
C	ACE	(A,E)
B	ABD	(A,D)
D	ADE	none
E	AE	none
A	A	none

Cliques

(Huang 236)

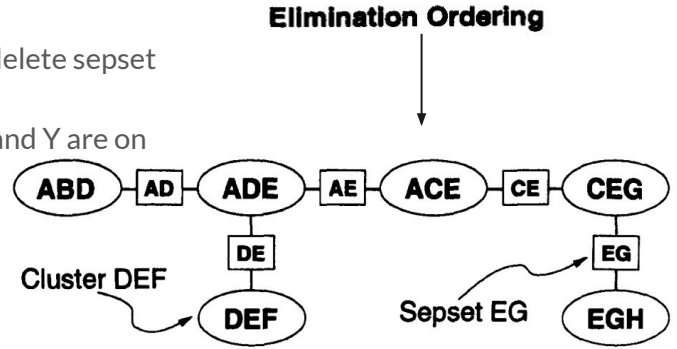
Elimination Ordering

Build Optimized Junction Tree

- Start with a forest of n trees each containing a clique, and an empty set *Petha*
- For each distinct pair of cliques X and Y:
  - Create a candidate sepset S_{xy}, labeled X intersection Y, with backpointers to the cliques
  - Insert S_{xy} into *Petha*
- Repeat until n-1 sepsets are in the forest:
  - Select a sepset S_{xy} from *Petha* following criteria (in code). And delete sepset from *Petha*
  - Insert the sepset S_{xy} between the cliques X and Y, but only if X and Y are on different trees in the forest.

Eliminated Vertex	Induced Cluster	Edges Added
H	EGH	none
G	CEG	none
F	DEF	none
C	ACE	(A,E)
B	ABD	(A,D)
D	ADE	none
E	AE	none
A	A	none

Cliques (Huang 236)



What’s left to do

- Propagator needs investigation
- Set up more test cases, and bug fix

References

Huang, Cecil, and Adnan Darwiche. “Inference in Belief Networks: A Procedural Guide.” *International Journal of Approximate Reasoning*, vol. 15, no. 3, Jan. 1996, pp. 225–263., doi:10.1016/s0888-613x(96)00069-2.