with this ordering:

Heuristically: i) move irrelevant terms outside innermost sum
ii) insert new term into product
iii) insert new term → prod.

ultimately :- $p(X_1|\underline{e}) = \frac{\phi(X_1,\underline{e})}{\sum_{x_1} \phi(X_1,\underline{e})}$

---

outcome of Elimin

Ex: Factors ϕ are general (local marginal / local cond / potential / intermediate eg $m(\cdot), m_i(\cdot)$)

- Ex: Each factor has scope ( )
- Queries, variables the.

Dealing with evidence

- evidence - non random variables (observed / clamped)
- Programming / implementation → reduce no. of operations via evidence potentials.

(*) Incorporate evidence into gm. via sum-product rule.
(*) Total evidence potential is merely a product; treat as addit. set of factors in GM representation

$$\tau(\underline{Y},\bar{e}) = \sum_{z,\underline{e}} \prod_{\phi \in F} \phi \times \partial(\underline{E},\bar{e})$$

- Elimin algorithm (W)(AG)(Q) ✓✓

PGM - visual prompts at suitable ordering

F

∂
∂
ϕ
ϕ
ϕ
ϕ

- "stack of factors"

(*) S-P-V-E — stack of factors; variables to be elimin

Ex: Input $(F, \underline{Z}, \prec)$ — ordering of $\underline{Z}$

- sum-product-variable-elim

(*) Eliminate one r.v. Z_i from set F (above subroutine)

- repeat continuously until all variables Z_i eliminated

(*) sum-product-eliminate var.

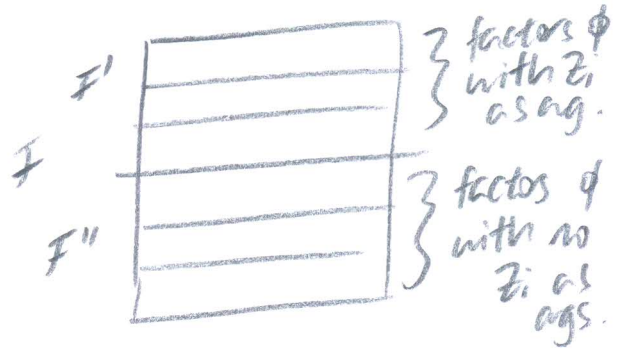
F // set of factors (aweve of potentials)

Z // variables to be eliminated

(*) Partition F into F' and F''

= depending on whether the factor ϕ contains Z as an argument

(*) F'' is the complement of F'

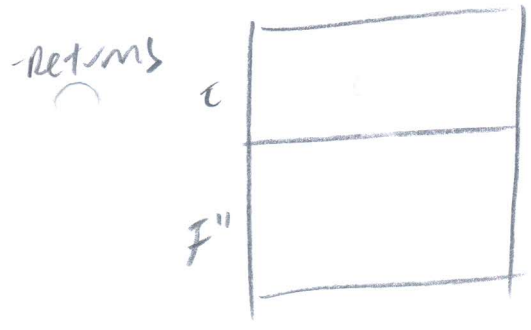


(*) ↘ corresponds to step where we put all terms that contain a particular variable in innermost summation

-(3)*) - Product

-(4)(*) - Sum

- $\tau(\cdot)$ does not contain Z_i



(*) Stack $|F_1| < |F_2| < \ldots$

(*) Normalise our product of remaining terms.

Ex: There may be multiple query r.v.s.

- Ex: An automated way of addressing all GMs (despite NP-hard)

-What are issues?

- Hard-wavy about ordering
- Irreducibility where factor is as big as model itself.

(*) Not guaranteed that factors are neatly identified.

- ordering may change size of factors (coupling)

Ex: move onto special cases

- complexity of variable elimination → concrete method for algo complexity

- (W)(A6): understand how this is done

(*) - c is subset of r.v.s captured by a particular factor that co-occurs with query x.

- HMM: factor size of 2

∴ complexity:- k^2

- (A7) (*) - This example

Ex: move away from chain models Ex: walkthrough elim. algorithm

- food web/complex network ; elimination in non-chain models

Query P(A|H)

- initial factor stack
- elimination order (not covered now)

mechanical recordy (not prob semant.)

- use graph too keep track of factors, newly formed factors

(*) - check you understood this example ✓

- After all terms eliminated ...

(W)(A8): step 8 - clarity. ✓

- understanding variable elimination

- Turn original graph → undirected moralised graph ✓

- Graph elimination - algebraic

- sequence of graph elimination ⇒ model with only query nodes

(*) New structures along the way giving meaningful graphical, algebraic info.

(through connection) ⟶ intermediate cliques ✓
cores. to algebraically eliminated

There is one-to-one corres of graph eliminants and algebraic eliminants (operational equivalence)

(*)- Allows visual inspection of largest clique ⇒ info about largest intermediate terms ✓

(*)- operationalisable way of determining inference complexity

---

Ex: Must take each elimination term in context.

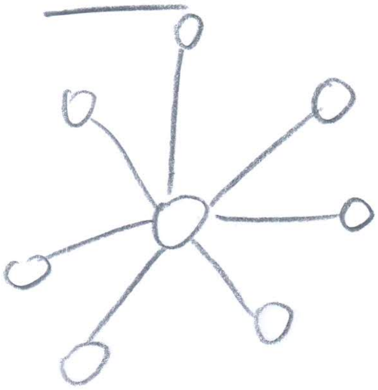
- clique tree
  - each graph eliminant can be connected by an edge when they share a subset of random variables.
- following elimination order ; ⇒ message passing along a clique tree
- form of message is a sum-product term

- Ex: message-passing as a general inference algo.
  (*) elimination is analogous to passing a message over a particular ordering (over clique trees).

(Q) where does elimination ordering come from
How does it affect algo complex.

- examples:- star and tree.

star



- can be DGM/UGM
- inference on star : how to do elimination of r.v.s?

- (A) "from degree 1 nodes"
- orderings (heuristic example)

- Ex: ~~not~~ Are we always able to find a good ordering that gives polynomial-time elimination algorithm

(W)(98) - we eliminate, therefore need to connect - need clarity on this aspect ✓

- Ising model → eg 100 x 100 pixel
  - can get cliques with 100 nodes.

Ex:
- However smart you are with elimination ordering; can still get exponentially large intermediate factors → still exponentially hard. ✓

(*) A pathway from elimination → message passing

(Q) If we decide to query another r.v. after graph elimin. do we have to redo?

- (*) minimum computation; redo messages (recompute a subset of messages)

Ex: Rose Tector

- Elimination - tractable inference on PGM by exploiting ~~via~~ elim. ordering; potential for max clique to be manageable
  - And can determine complexity precisely before working on it

Appn.
- messages can be bi-directional