Key idea:

A "simple" graph has a variable ordering which results in:

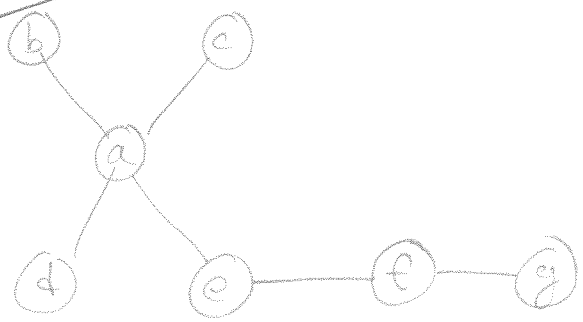
(1) fewer edges when eliminating nodes (graph story)

(2) fewer nz's when doing G.E. (LinAlg story).

Which ordering is best?

- determining best order is NP-hard!
  - > harder than solving the original problem!
- Good heuristics?
  - > Minimum degree ordering.
  - > COLAMD (faster approx of MDO)
- Computing ordering takes time.
  - > Balance quality of ordering vs. time needed to compute it.
  - > MDO outperforms COLAMD on most SCAM problems...

EXAMPLE

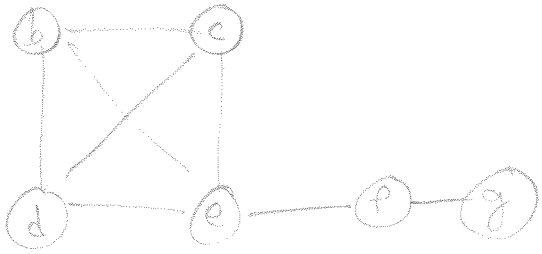


adjacency matrix

	a	b	c	d	e	f	g
a	X	X	X	X	X		
b	X	X					
c	X		X				
d	X			X			
e	X				X	X	
f					X	X	X
g						X	X

19

eliminate a



	b	c	d	e	f	g
b	X	X	X	X		
c	X	X	X	X		
d	X	X	X	X		
e	X	X	X	X	X	
f				X	X	X
g					X	X

22

(etc)... We can measure "hardness" by counting the # of edges.

Try another ordering: g f e d c b a

	g	f	e	d	c	b	a
g	X	X					
f	X	X	X				X
e		X	X				
d				X			X
c					X		X
b						X	X
a		X	X		X	X	X

19

Factor Graphs

Bayes nets aren't quite our SLAM graphs...

$$p(abc) = p(a)\,p(b|a)\,p(c|ab)$$

un-natural & neither a motion model or obs model.

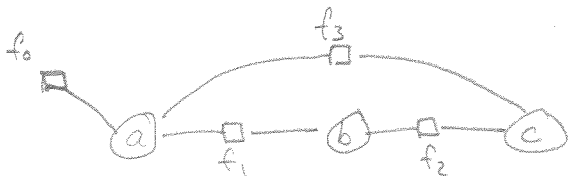
ASIDE

factor graphs can represent Bayes net structures, plus others

bipartite graph.

$\Rightarrow$ A second type of graphical model

Joint is the product of "factors"



$$p(abc) = f_0(a)\,f_1(a,b)\cdot f_2(b,c)\cdot f_3(a,c)$$

$$(\text{eg.}) \quad = p(a)\,p(b|a)\,p(c|b)\,p(c|a)$$

In general, we are not concerned that the $\int \prod f() = 1$... just that it has the right "shape".

$\Rightarrow$ constant factors are not that interesting!

★ Key: maximum likelihood solution found by maximizing product of factors... or equiv, sum of log factors.

Sound familiar? $\Rightarrow$ least squares!

SLAM graph

A stylized factor graph: In SLAM, almost all factor nodes are degree 2. $\Rightarrow$ we omit the boxes.

SLAM Graph: Nodes = variables
Edges = factors
Edge direction = meaningless,

Factor Graphs

With factor graphs, we eliminate one variable at a time in order to solve the system for one var (the "last" one) then backsolve for the earlier variables.

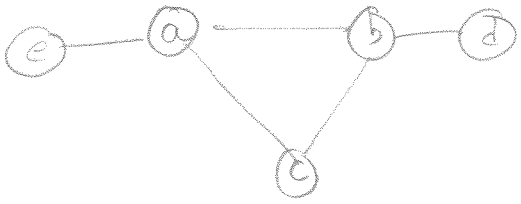
⇒ eliminate b ⇒

↖ some new factor, a func. of f_1 & f_2

Similar to Bayes nets, but there is no difference between children/parents. All get connected.

Key: When eliminating a node, we form a clique between its neighbors.

Some elimination orders create more complex factor graphs than others.



Consider elimination order a, b, c, d, e vs. e, d, c, b, a.

Sparsity

- It's actually more subtle than this!
  - Need to be able to solve $A_{sx} = b$ such that intermediate steps are also sparse!
  - E.g. if we use cholesky, LL^T should be sparse.

Consider info. matrix

$$\begin{bmatrix} 16 & 8 & 4 & 1 \\ 8 & 10 & 0 & 0 \\ 4 & 0 & 20 & 0 \\ 1 & 0 & 0 & 8 \end{bmatrix}$$

(nz = 6) pretty sparse!

(do symbolic cholesky / Gauss. Elimination)

⇒ lots of fill-in, even though A was sparse.

Now consider

$$\begin{bmatrix} 8 & 0 & 0 & 1 \\ 0 & 20 & 0 & 4 \\ 0 & 0 & 10 & 8 \\ 1 & 4 & 8 & 16 \end{bmatrix}$$

(repeat Chol, GE)

⇒ Sparsity preserved!

How are these matrices related?

- Simple transform via "permutation matrix" $P = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$

$$A' = PAP^T$$

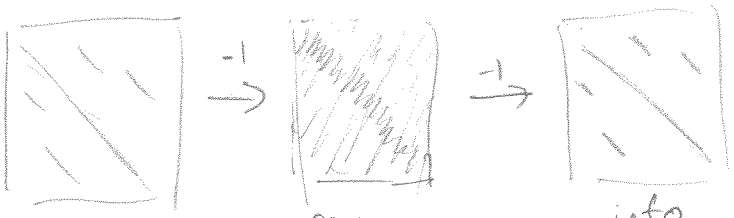
- Or, more simply, we wrote the state variables in a different order.

How does marginalization affect sparsity?

- Suppose we have a big sparse $J^T J$.
  (information matrix)
- Will its inverse be sparse?

  > in general, no.

Key: "info matrices can be sparse, cov. matrices seldom."

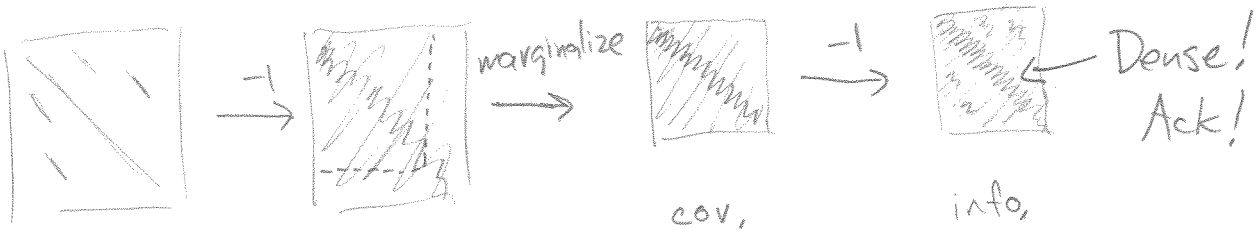


info. (sparse)

cov. (dense, though SPD)

same, of course!

How do we marginalize? Drop rows/cols of cov...



- Marginalization destroys sparsity of info matrix.

  We have a big tension now for speed:
  - Do we marginalize-out old info to keep matrices small?
  - Or do we keep everything & hope that sparse matrix methods will "save" us?