PGMs week 5 Inference: The Junction-Tree Algorithm

- **WATCH** the Koller videos on Variable Elimination and Clique Trees:
  - *07.5 - Inference: Variable Elimination: Graph-Based Perspective*
  - *07.6 - Inference: Variable Elimination: Finding Elimination Orders*
  - *PGM37 - Message Passing: Clique Tree Algorithm Correctness*
  - *PGM38 - Message Passing: Clique Tree Algorithm Computation*
  - *PGM39 - Message Passing: Clique Tree and Independence*
  - *PGM40 - Message Passing: Clique Tree and VE*
- Notes on these videos:
  - What is elsewhere known as a ‘Junction Tree’, Koller calls a ‘Clique Tree’, a very apt name since it is a tree of the maximal cliques.
  - Variable elimination plays a key role in determining the graph structure. Different elimination orders can result in different induced graphs, with possible major differences in computational complexity.
  - Koller also uses the variable elimination procedure to link the various nodes in the graph. See lower down for an (possibly easier) alternative.
  - Each sep-set separates the resulting tree in two parts with no paths linking them except through this sep-set.
  - The message passing algorithm Koller uses here is also known as the Shafer-Shenoy algorithm. It changes very little from the one that we have used for Cluster Graphs and therefore is quite intuitive.
  - With an appropriate message schedule we only need to pass a message once in both directions over each link to achieve convergence to the true/correct marginal values.
- Read Barber chapter 6: The JT algorithm.
  - The focus here is on the absorption/Hugin message passing algorithm, with the Shafer-Shenoy algorithm mentioned as an alternative in the later parts of the chapter.
  - Building the Junction Tree is also described somewhat differently (more like the summary below).
- A summary of the procedure to build the Junction Tree (Michael Jordans version):
  - If starting from a Bayes Net, first moralise it and then remove the arrows.
  - Use Variable Elimination to ‘triangulate’ the graph, resulting in an ‘induced’ graph. (Note that the concept ‘triangulate’ is somewhat more subtle than its name seems to imply.)
  - Then find the maximal cliques in the induced graph, the variables in each such a clique are combined in a (compound) node. (You can automate this via the Bron-Kerbosch algorithm.)
  - Each potential from the original graph is allocated to exactly one of these clique nodes – it does not matter which as long as it contains all the variables needed for the original potential. The clique node potential is the product of all the potentials allocated to it. If none are allocated, the clique has an implicit unity potential.
  - We can link these nodes by finding a maximal spanning tree over the clique nodes (you can use Prims algorithm, Kruskall’s algorithm, or the elimination messaging system of Koller for this):
    - ∗ Begin with no edges.
    - ∗ At each step add an edge between the two clique nodes that shares the largest sep-set and will not cause a loop between clique nodes.
    - ∗ Repeat this until all the nodes are connected.

- $*$ Easy checks:
  - · In the *induced* tree, identify the maximal cliques. Check that for each pair of such linked cliques, the sep-set common to them separates the graph into two parts. There should be no other paths linking them apart from those running through this sep-set.
  - · In the *Clique Tree*, check that the RIP holds.
  - · There should be no loops in the the Clique Tree.

- Exercise: Use the variable naming convention from the previous tasks for the Hamming (7,4) code i.e. check bits: b_4, b_5, b_6, the three message bits each only shared by two parity checking potentials: b_1, b_2, b_3, and the message bit shared by all the parity checking potentials: b_0.
  - Determine the induced graphs for the following elimination orders:
    1. $b_4, b_5, b_6, b_1, b_2, b_3, b_0$
    2. $b_1, b_2, b_3, b_4, b_5, b_6, b_0$
    3. $b_0, b_1, b_2, b_3, b_4, b_5, b_6$
  - Compare these in terms of the size of the largest clique and the effect that this has on computational requirements. The result using elimination order 1 is given in Figure 1.
  - Construct the Clique Trees for the above triangulations. The result using elimination order 1 is given in Figure 2.
  - Specialise the Shafer-Shenoy message passing system you implemented in the previous task to the Junction Tree you created here. I.e use a proper message-passing schedule that only sends a message when all the incoming messages it relies on are finalised. And do only one pass in each direction of each link.
  - On one (or more) of these, run the Shafer-Shenoy algorithm to find the marginal beliefs. Compare the resultant decodings to that obtained from the full joint, as well as those from using the Shafer-Shenoy algorithm on a loopy graph (last week's task).

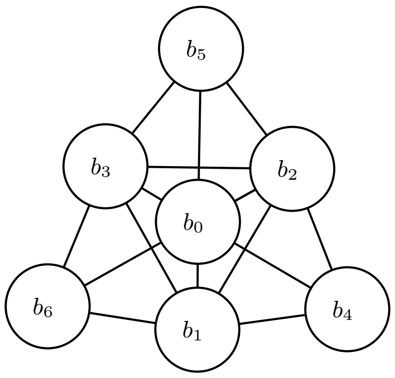

Figure 1: Induced graph for Hamming (7,4) error correcting code using elimination order $b_4, b_5, b_6, b_1, b_2, b_3, b_0$.

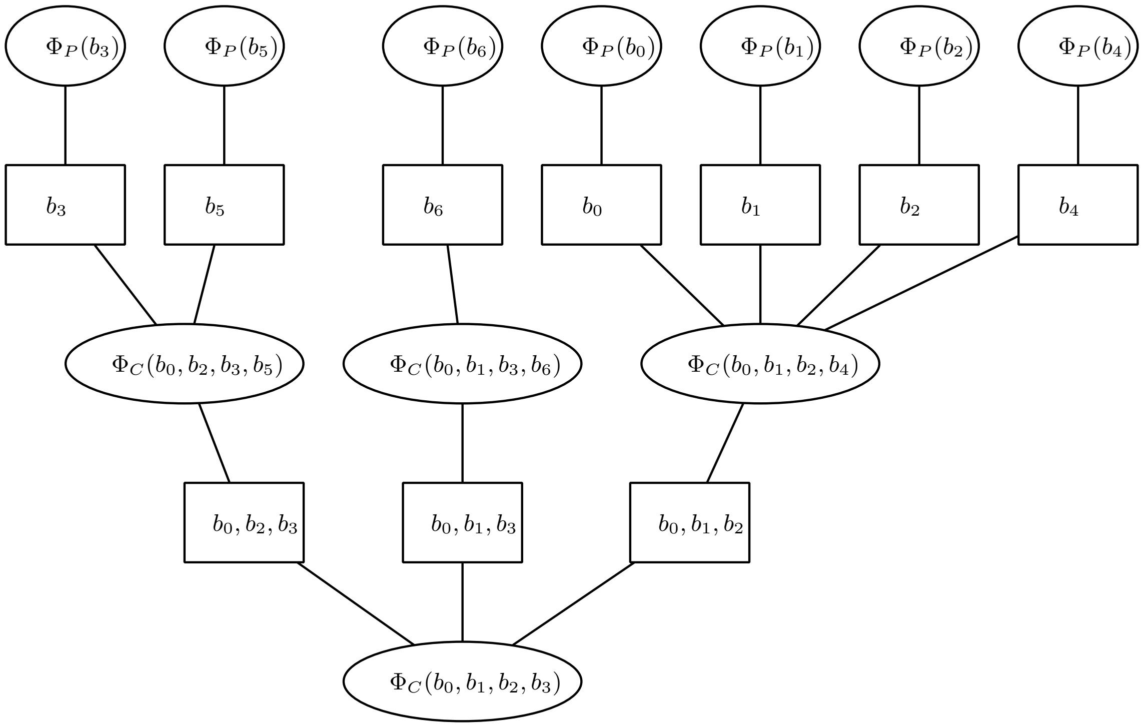

Figure 2: Junction Tree for Hamming (7,4) error correcting code using elimination order $b_4, b_5, b_6, b_1, b_2, b_3, b_0$.