Bayesian nets
Meeting notes 1/13/2023

Everybody was there. Next week we'll start at 2:00 rather than 2:30.

We started with 3 'easy things':

(a) Amber read her first draft of an abstract on the project for UROP, which was very complete. The basic idea was all there—using Bayesian nets to model scientific inference in terms of scientific theory change, starting with a structurally adaptive model attempting to capture the causal pattern of a 'world' in terms of a stream of evidence. I'll ask that she make the final copy available to all of us, for further use.

(b) second easy thing—showing Patrick how not merely to load the program but to make it run. We're working in Visual Studio Code (downloadable free), with the main code accessed by a Jupyter notebook. I think we should all make ourselves able to read and work with the program. November 30th version (perhaps with Dennis's addition of quotation marks) attached.

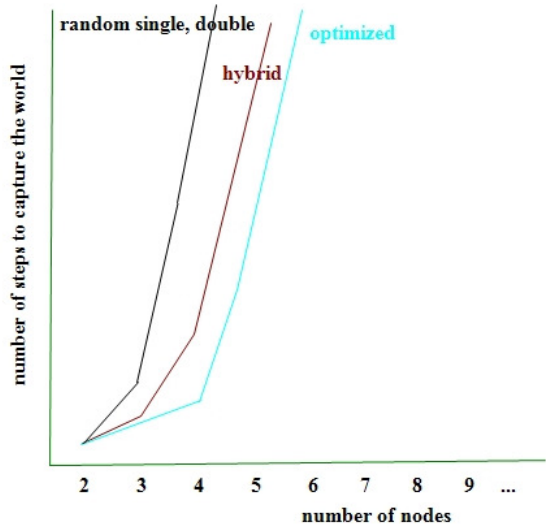
(c) third easy thing—figuring out the 'sample space' of networks with n nodes, where we don't allow loops but do allow 'no contact' or non-linked nodes. This is primarily in Zhongming's hands. Patrick outlined a strategy that attempted to figure out the number of networks for n nodes by building on an assumed answer for n-1 nodes, starting with 2 nodes (a->b, b->a, and no link between a and b). But by the end of the meeting Zhongming seemed to have a more promising approach in terms of n nodes and options for 1 arrow between them, 2 arrows between them, and so forth.

We figure the sample space is going to explode with increasing n, but we don't know exactly how. But knowing that will be very relevant to what Amber presented, which was a start on the graph idea from last time. We have a number of 'network-change' or mutation heuristics on hand: simple point genetic algorithm
Double point genetic algorithm
Randomly single-point or double-point ga
Hybridization of top performers
'keep the good parts' versions of these
'don't make the same mistake twice' versions of these.
..and I may have left some out.

The graph idea: construct graphs of how many steps different mutation heuristics take to find the 'world' for networks of increasing numbers of nodes. Something like the picture below.
What this calls for is running a number of cases in which (a) we start with a random network, iterate out mutation algorithm, and count steps to matching the world, and (b) averaging over that sample.

The problem Amber indicated was just the 'start with a random network' part. Pyagrum won't generate a 'looped' network, but that means it may take time to get it to generate a random

network to begin with. Our genetic algorithm mutation, moreover, may make a looped network from something that wasn't, compounding the problem.



This is the 'it would look something like this' graph. Amber actually had a real one for a couple of strategies.

One suggestion: to build a 6-node network, first build a 2-node and check that it's not looped. If it's not, add a node and check if that's looped. If it is, try again. If it isn't, add a further node. It has yet to be seen if this helps with computation time at all.

A further graph idea is to track a single network changing with one or more of these strategies, and to see wat the pattern of change is, measured in terms of approximation to 'the world.' Is it characteristically a gradual increase in approximation, or does it go through periods of slow growth followed by dramatic 'leaps,' in the form of punctuated equilibrium? And does that differ with different network re-wiring heuristics?

Some of the strategies may also be more psychologically or sociologically realistic. The 'keep the good parts' seems particularly promising in that respect. If your theory seems to work on 80% of the evidence, wouldn't you keep that and try to tinker with the 20% it gets wrong?

For next week, starting at 2:00:

Dennis is going to have the hybridization strategy in hand. We agreed that the best way was probably: Pick a top two performing networks. Choose a random assortment of spots in the best one's code. Keep those, and substitute the values in the other spots from the second best.

This will face the 'is there a loop?' problem again. That seems a bug-a-boo. I wonder if there is an alternative to relying on PyAgrum there or in our network construction.

But Dennis also mentioned a nice wrinkle on this: Keep more of the spots of the more successful strategy. In fact one could keep a number of spots of network A and network B depending on the relative success of A and B.

Dennis is also working on a way to code our 'information flow' which reflects the stages at which particular nodes are activated, as a direct response to a difficulty that Amber noted previously that two different representation networks could come up 'perfect' because we weren't distinguishing between the order at which different nodes were activated.

Amber is going to keep working on the graph problem, and we will undoubtedly here more about the non-looped random network generating issue next time.

Zhongming is going to keep working on calculating the sample space of qualifying networks given a certain number of nodes.