Manual for the quantum causal discovery MatLab code

Christina Giarmatzi[1,2] and Fabio Costa[1]

[1] *Centre for Engineered Quantum Systems, School of Mathematics and Physics, University of Queensland, Brisbane, QLD 4072, Australia*

[2] *Centre for Quantum Computer and Communication Technology, School of Mathematics and Physics, University of Queensland, Brisbane, QLD 4072, Australia*

Input to the code - your own inputs

Three files are required as an input to the code. If you simply want to test the code with arbitrary causal structures, you can generate these files with the attached Mathematica file. If you want to discover the causal model of a particular process matrix, here is what you should do. The process matrix should be stored in a .dat file. Other file types that MatLab understands as a matrix should work too, but you have to change the file type in the Imports section (second section). The second file required is a matrix, again stored in a .dat file, that contains the dimensions of input and output system of the parties. For example, for parties {1, 2, 3, 4} with dimensions of input and output systems d_I^n, d_O^n respectively for the party n, it would be the following.

$$\begin{matrix} d_{1_I} & d_{1_O} \\ d_{2_I} & d_{2_O} \\ d_{3_I} & d_{3_O} \\ d_{4_I} & d_{4_O} \end{matrix}$$

The third file is a script, so a .m file, that contains information about the output subsystems of the parties. If there are no subsystems it can be empty, but it still needs to be created and imported. When it's not empty, for parties that have output subsystems for example $\{2, 4\}$ with dimensions $d^1_{2_O} = 2, d^2_{2_O} = 3, d^3_{2_O} = 3, d^1_{4_O} = 3, d^2_{4_O} = 4$, with the superscript denoting the numbering of the subsystems, the script should write: `subdim{2} = [2 3 3]; subdim{4} = [3 4];` After these files are created, they should be named correctly: W[tag].dat, dmat[tag].dat and subdim[tag].m, with [tag] being a name tag for this process.

Input to the code - inputs from Mathematica

To test the code, you can use the Mathematica notebook provided. The instructions how to generate the files are in the notebook itself. You should have three files generated: W[tag].dat, dmat[tag].dat, subdim[tag].m., where [tag] is the name chosen for this set of inputs.

Output of the code

Before you run the code there are two variables you need to specify in the very first section: the variable 'tag' which would be the tag you chose, and the variable 'folder' which is the path in which the above three files are stored (remember to add a backslash \ at the end of the path).

Now you're ready to run the code. On the command window you should see what is shown in the table below. The output of the code is the left part. I have added comments on the right part to enhance the understanding of each output of the code. These comments are only shown here, it is not an output of the code. As an example, a Markovian process is used with parties {1, 2, 3, 4} and causal order $\{2 \prec \{3, 4\} \prec 1\}$ and a causal model as shown in the diagram below.

For the full causal model, one needs the exact mechanisms, namely the channels and the specific input states of the first parties. These can be accessed as follows.

For the input states, they are stored in the form `term_input{x}`, where `x` is a party that belongs to the first set, i.e. {2} in the above example.

For an arrow, in which the receiving party has only one incoming arrow and the associated output system is not a subsystem, it is stored as `term{x,y}`, for the arrow from party `x` to party `y`. Also, the last parties are stored like that, with `y` being $N+1$, where N is the total number of parties. In our example there is one term of that type, which is `term{1,5}` because party 1 is last and $N = 4$. This term is the identity matrix with the dimension of the output system of that party.

For an arrow, in which the receiving party has only one incoming arrow and the associated output system is a subsystem, it is stored as `subterm{x, y, z}`, for the arrow from subsystem `z` of party `x`, to party `y`. In our example it would be: `subterm{2,4,1}` and `subterm{2,3,2}`.

For arrows where the parent space is more than one party, meaning that there is more than one incoming arrow to the receiving party, the overall channel from the parent space of party `x` to party `x` is stored as `subterm_tot{x,1}`. In our example, those are `subterm_tot{1,1}` being a channel from the output of parties {2, 3, 4} to the input of 1.

```
the_sets =                                  The non-signalling sets, the bottom line is the first set, then the second, etc.
   1   0
   3   4
   2   0

Time 0.4487                                 The time it took to evaluate the above.
                                                                                The clock resets here.
                                                                                What follows is the primal arrows.
Link from party 3 to party 1                                                    Depicted in the graph.
Link from party 4 to party 1                                                    Depicted in the graph.
Link from subsystem 2 of party 2 to party 3     Depicted in the graph without the information on the subsystem 2.
Link from subsystem 1 of party 2 to party 4     Depicted in the graph without the information 1.

4 primal arrows                                                                 The number of primal arrows

primal_arrows =                             A compact list of the arrows, without any information about the subsystems.
   3   1                                                                        from 3 to 1
   4   1                                                                        from 4 to 1
   2   3                                                                        from 2 to 3
   2   4                                                                        from 2 to 4
Time 0.63989                                                                    The time it took to evaluate the above.
                                                                                Clock resets again.
                                                                                What follows is the secondary arrows.
Link from subsystem 3 of party 2 to party 1     Depicted in the graph without the information 3.

1 secondary arrows                                                              The number of secondary arrows

secondary_arrows =                                                              A compact list of the arrows.
   2   1                                                                        from 2 to 1

Time 0.083953                                                                   The time it took to evaluate the above.

The process is Markovian
```

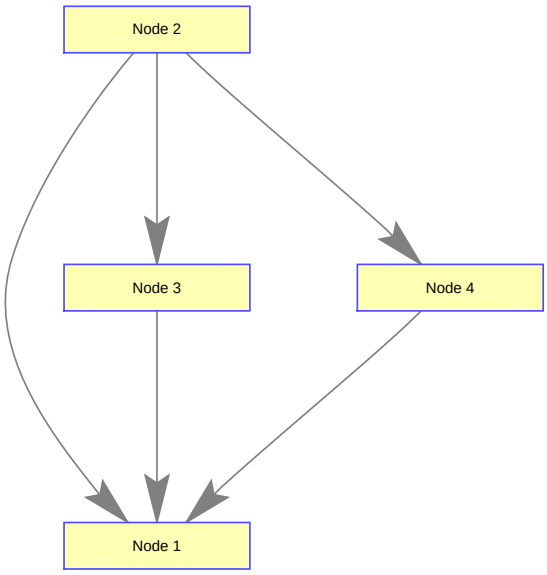

Figure 1: The code outputs this DAG, given the particular example of a Markovian process matrix.