npj | Quantum Information

www.nature.com/npjqi

ARTICLE OPEN

A quantum causal discovery algorithm

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

Finding a causal model for a set of classical variables is now a well-established task—but what about the quantum equivalent? Even the notion of a quantum causal model is controversial. Here, we present a causal discovery algorithm for quantum systems. The input to the algorithm is a process matrix describing correlations between quantum events. Its output consists of different levels of information about the underlying causal model. Our algorithm determines whether the process is causally ordered by grouping the events into causally ordered non-signaling sets. It detects if all relevant common causes are included in the process, which we label Markovian, or alternatively if some causal relations are mediated through some external memory. For a Markovian process, it outputs a causal model, namely the causal relations and the corresponding mechanisms, represented as quantum states and channels. Our algorithm opens the route to more general quantum causal discovery methods.

npj Quantum Information (2018)4:17 ; doi:10.1038/s41534-018-0062-6

INTRODUCTION

The discovery of causal relations is a basic and universal task across all scientific disciplines. The very nature of causal relations, however, has been a long-standing subject of controversies with the central question being what, if anything, distinguishes causation from correlation.

It is only recently that a rigorous framework for causal discovery has been developed.[1,2] Its core ingredients are *causal mechanisms* that are responsible for correlations between observed *events*, with the possibility of external *interventions* on the events. It is the possibility of interventions that provides an empirically well-defined notion of causation, distinct from correlation: an event A is a cause for an event B if an intervention on A results in a change in the observed statistics of B. A causal model is typically defined as a set of direct-cause relations and a quantitative description of the corresponding causal mechanisms. The causal relations are represented as arrows in a graph and the causal mechanisms are usually described in terms of transition probabilities (Fig. 1).

Among the most important achievements of causal models is the development of algorithms for causal discovery. The objective of such algorithms is to infer a causal model based on observational and interventional data. Such algorithms have found countless applications and constitute one of the backbones in the rising field of machine learning, as causal discovery is a crucial intermediate step for many of its algorithms.

It is a natural question whether similar algorithms can be developed for quantum systems. In simple quantum experiments, causal relations are typically known and well under control. However, the fast growth of quantum technology goes towards the development of networks of increasing size and complexity. Hence, appropriate tools to recover causal relations might become necessary for the functioning of large, distributed quantum networks, as it is already the case for classical ones.[3] Causal discovery might further detect the presence of "hidden common causes", namely external sources of correlations that might introduce systematic errors in a quantum experiment or device. Finally, from a foundational perspective, the possibility of discovering causal relations from empirical data opens the possibility to recover causal structure from more fundamental primitives.

Classical causal discovery algorithms, however, fail to discover causal relations in quantum experiments.[4] A considerable effort has been recently devoted to solve this tension and transfer causal modeling tools to the quantum domain,[5–14] leading to the formulation of a quantum causal modeling framework.[15,16] (See refs. [17,18] for a broader philosophical context.)

Here we introduce an algorithm for the discovery of causal relations in quantum systems. The starting point of the algorithm is a description of a quantum experiment (or "process") that makes no prior assumption on the causal relations or temporal order between events.[19] Given such a description, encoded in a *process matrix*, the algorithm extracts different levels of causal information for the events in the process. It determines whether or not they are *causally ordered*, namely if they can be organized in a sequence where later events cannot influence earlier ones. If a causal order exists, the algorithm finds whether all common causes are modeled as events in the process matrix—a property expressed by the condition of *Markovianity*, as defined in ref. [15]. If the process is Markovian, the algorithm outputs a causal model for it: a causal structure (depicted as arrows connecting events) together with a list of quantum channels and states that generate the process.

The complexity of our algorithm scales quadratically with the number of events, although the size of the problem itself (the dimension of the process matrix) is exponential. This suggests that the algorithm can be used efficiently given an efficient encoding of the input to the code. We further comment on possible extensions of the algorithm to deal with processes that are not Markovian, not causally ordered, or that follow a different definition of Markovianity.[16] We provide the implementation of the algorithm, written on MatLab, some examples, and a Manual (https://github.com/Christina-Giar/quantum-causal-discovery-algo.git). The code uses some functions developed by Toby Cubitt (http://www.dr-qubit.org/matlab.html).

[1]Centre for Engineered Quantum Systems, School of Mathematics and Physics, University of Queensland, Brisbane QLD 4072, Australia and [2]Centre for Quantum Computation and Communication Technology, School of Mathematics and Physics, University of Queensland, Brisbane QLD 4072, Australia
Correspondence: Christina Giarmatzi (c.giarmatzi@uq.edu.au)

Received: 17 April 2017 Revised: 8 December 2017 Accepted: 19 January 2018
Published online: 06 March 2018

RESULTS

Process framework

We will use a formulation of quantum mechanics that can assign probabilities to quantum events with no prior knowledge of their causal relations.[19] This formulation is based on the "combs" formalism for quantum networks,[20] with the main difference that the causal order between events is not assigned in advance.

In this framework, a *quantum event* A can be thought to be performed by a party inside a closed laboratory (Fig. 2)—which is associated with an input and an output Hilbert space, $\mathcal{H}^{A_\mathrm{I}}$ and $\mathcal{H}^{A_\mathrm{O}}$, respectively—and is represented by a completely positive (CP) map $\mathcal{M}^{A_\mathrm{I}\to A_\mathrm{O}}:\mathcal{L}(\mathcal{H}^{A_\mathrm{I}})\to\mathcal{L}(\mathcal{H}^{A_\mathrm{O}})$, where $\mathcal{L}(\mathcal{H}^{S})$ is the space of linear operators over the Hilbert space of system S. A *quantum instrument* is the collection of CP maps $\mathcal{J}^A=\{\mathcal{M}^A\}$, such that $\sum_{\mathcal{M}^A\in\mathcal{J}^A}\mathcal{M}^A$ is a CP and trace-preserving (CPTP) map.

It was found that, for a set of parties $\{A^1,\ldots,A^n\}$, the joint probability of their CP maps to be realized, given their instruments, is a function of their maps and some matrix that mediates their correlations:

$$p\left(\mathcal{M}^{A^1},\cdots,\mathcal{M}^{A^n}|\mathcal{J}^{A^1},\cdots,\mathcal{J}^{A^n}\right)=\mathrm{Tr}\left[W^{A_\mathrm{I}^1A_\mathrm{O}^1\cdots A_\mathrm{I}^nA_\mathrm{O}^n}\left(M^{A_\mathrm{I}^1A_\mathrm{O}^1}\otimes\cdots\otimes M^{A_\mathrm{I}^nA_\mathrm{O}^n}\right)\right]. \tag{1}$$

Using a version of the Choi–Jamiolkovsky (CJ) isomorphism,[21,22] the CJ matrix $M^{A_\mathrm{I}A_\mathrm{O}}\in\mathcal{L}(\mathcal{H}^{A_\mathrm{I}}\otimes\mathcal{H}^{A_\mathrm{I}})$, isomorphic to a CP map $\mathcal{M}^A:\mathcal{L}(\mathcal{H}^{A_\mathrm{I}})\to\mathcal{L}(\mathcal{H}^{A_\mathrm{O}})$ is defined as $M^{A_\mathrm{I}A_\mathrm{O}}:=[\mathcal{I}\otimes\mathcal{M}(|\phi^+\rangle\langle\phi^+|)]^\mathrm{T}$, where $\mathcal{I}$ is the identity map, $|\phi^+\rangle=\sum_{j=1}^{d_{A_\mathrm{I}}}|jj\rangle\in\mathcal{H}^{A_\mathrm{I}}\otimes\mathcal{H}^{A_\mathrm{I}}$, $\{|j\rangle\}_{j=1}^{d_{A_\mathrm{I}}}$ is an orthonormal basis on $\mathcal{H}^{A_\mathrm{I}}$ and T denotes matrix transposition in that basis and some basis of $\mathcal{H}^{A_\mathrm{O}}$. Finally, $W^{A_\mathrm{I}^1A_\mathrm{O}^1,\cdots,A_\mathrm{I}^nA_\mathrm{O}^n}\in\mathcal{L}\left(\mathcal{H}^{A_\mathrm{I}^1}\otimes\mathcal{H}^{A_\mathrm{O}^1}\otimes\cdots\otimes\mathcal{H}^{A_\mathrm{I}^n}\otimes\mathcal{H}^{A_\mathrm{O}^n}\right)$ is a positive semi-definite matrix that lives on the combined Hilbert space of all input and output systems of the parties and is called *process matrix*. Equation (1) can be seen as a generalization of the Born rule, and the process matrix as a generalization of the quantum state, as it is the resource that allows calculating joint probabilities for all possible events. Just as the Born rule is the only non-contextual probability assignment for POVM measurements,[23] Eq. (1) is the only non-contextual probability rule for CP maps.[24]

Here we are interested in situations where causal relations define a partial order, which we call *causal order*. We identify causal relations with the possibility of signaling: if the probability of obtaining an outcome in laboratory B can depend on the settings in laboratory A, we say that A *causally precedes* B, and write $A\prec B$. (We write $A\|B$ if no signaling is possible and say A and B are *causally independent*.) The process matrices that define a causal order between the events are called *causally ordered*.

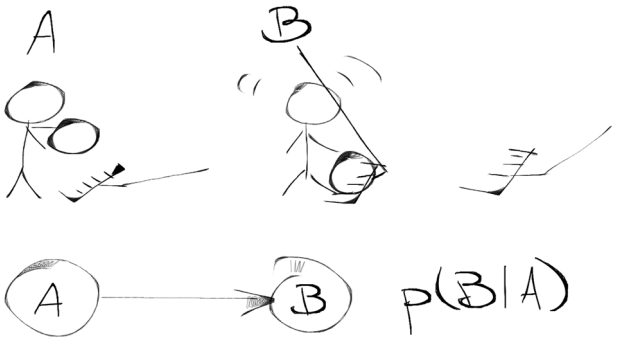

Fig. 1 Causal relations. An example of a causal relation and its representation in a graph. Credits: C. Giarmatzi

From mathematical to graphical representation

The causal structure encoded in the process matrix can be represented by a *directed acyclic graph (DAG)*. A directed graph is a pair $\mathcal{G}=\langle\mathcal{V},\mathcal{E}\rangle$, where $\mathcal{V}=\{V_1,\ldots,V_n\}$ is a set of *vertices* (or nodes) and $\mathcal{E}\subset\mathcal{V}\times\mathcal{V}$ is a set of ordered pairs of vertices, representing *directed edges*. A *directed path* is a sequence of directed edges where, for each edge, the second vertex is the first one in the next edge. Figure 3(a) shows a directed path from V_1 to V_3. A *directed cycle* is a directed path that ends up in a vertex already used by the path, as shown in Fig. 3(b). A DAG is a directed graph with no directed cycles, as shown in Fig. 3(c). We refer to edges as *causal arrows*.

Following ref. [15], we define a quantum causal model by associating a specific type of process matrix to a DAG. To this end, we associate a party, with input and output spaces, to each node of the DAG. If the node has more than one outgoing arrow, then the output space is composed of subsystems, with one subsystem for each arrow. We refer to them as *output subsystems*. We define the *parent space* Γ^A of a node A as the tensor product of all output subsystems associated with an arrow ending in A. A *Markov quantum causal model* is then defined by a collection of quantum channels, one for each node A, connecting the parent space of A to its input space.

Now let us see how a process matrix whose causal structure is represented by a DAG looks like. It will be a tensor product of three types of factors: input states for the set of parties with no incoming arrow in the DAG, channels connecting each input system of a remaining party with its parent space, and finally the identity matrix $\mathbb{1}$ for the output systems of the set of parties with

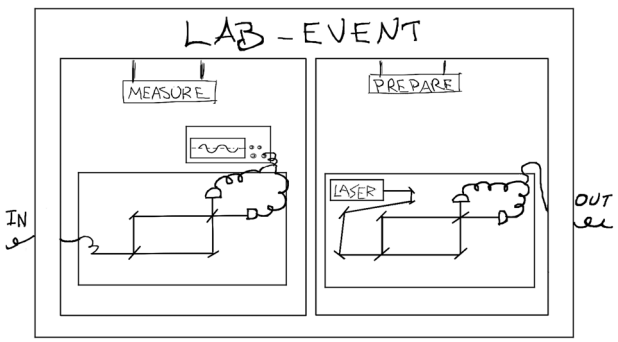

Fig. 2 Lab-event. A picture of a quantum event, consisting of a measurement stage of the input system and a preparation stage for the output system. It may also be simply a unitary transformation. Credits: C. Giarmatzi

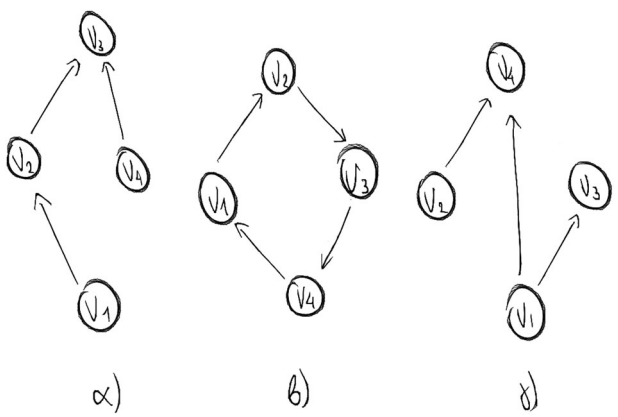

Fig. 3 Examples. Figure (**a**) shows a DAG with a directed path from V_1 to V_3, (**b**) an example of a directed cycle and (**c**) another example of a DAG. Credits: C. Giarmatzi

no outgoing arrows in the DAG. For example, if $\{F^1, F^2, \ldots, M^1, M^2, \ldots L^1, L^2, \ldots\}$ is a set of parties where F, M and L is the label for the three set of parties described above (first, middle, and last), respectively, then their process matrix would be

$$W^{F_{\mathrm{I}}^1 F_{\mathrm{O}}^1 \cdots} = \rho_1^{F_{\mathrm{I}}^1} \otimes \rho_2^{F_{\mathrm{I}}^2} \otimes \cdots T^{\Gamma^{M_{\mathrm{I}}^1} M_{\mathrm{I}}^1} \otimes T^{\Gamma^{M_{\mathrm{I}}^2} M_{\mathrm{I}}^2} \otimes \ldots \mathbb{1}^{\mathbb{L}_{\mathrm{O}}^1 \mathbb{L}_{\mathrm{O}}^2 \cdots}, \tag{2}$$

where $T^{\Gamma^{M_{\mathrm{I}}^j} M_{\mathrm{I}}^j}$ is a matrix representing a CPTP map $\mathcal{T}$ from the parent space of M_{I}^j, $\Gamma^{M_{\mathrm{I}}^j}$, to the input of M_{I}^j, M_{I}^j, via the isomorphism (same as the one used to describe the CP maps of the parties, but without transposition). $T^{\Gamma^{M_{\mathrm{I}}^j} M_{\mathrm{I}}^j} := \mathcal{I} \otimes \mathcal{T}(|\phi^+\rangle\langle\phi^+|) \in \mathcal{H}^{\Gamma^{M_{\mathrm{I}}^j}} \otimes \mathcal{H}^{M_{\mathrm{I}}^j}$.

From now on we identify a channel with its matrix representation. A representation of Markovian processes as in Eq. (2) is also employed in the study of open quantum systems.[25]

The above condition for the causal structure of the process matrix to be described by a DAG is a quantum generalisation of the Markov condition for classical variables and so it can be called the quantum Markov condition.[15] (We will comment below on a slightly different possible definition.[16]) Such a process matrix is also causally ordered, with a partial order defined by the DAG. However, the class of causally ordered process matrices is strictly broader than Markovian ones, and they are represented by quantum combs.[20] As we will see later, causally ordered processes that are not Markovian can be understood as processes involving correlations with some unobserved systems—called 'latent' variables. The algorithm we present here detects whether a process matrix is causally ordered and if it is, it outputs the causal order of sets of parties that are causally independent. It further detects Markovianity and for a Markovian process it outputs the DAG associated with the process matrix. We discuss in section "Non-Markovian processes" possible extensions of the algorithm that could output a DAG for a non-Markovian process.

Quantum causal discovery

The code takes as an input a process matrix, which can be obtained from experimental data. The procedure is similar to quantum state tomography: one can reconstruct the process matrix given the probabilities arising from informationally complete instruments.[15] The code also requires information about the dimension of all input and output systems and the decomposition of the output systems into subsystems. The output subsystems represent possible outgoing arrows, to be discovered by the algorithm.

The linear constraints. A process matrix of the form of Eq. (2) satisfies a set of linear constraints. This set identifies a DAG—in fact, each constraint corresponds to a particular element in the DAG. There are two types of constraints.

Open output: A party A has an *open output* when in the process matrix W there is an identity matrix on the corresponding output system A_{O}. This translates to the following linear constraint:

$$\widetilde{\mathbb{1}}^{A_{\mathrm{O}}} \otimes \mathrm{Tr}_{A_{\mathrm{O}}} W = W, \tag{3}$$

where $\widetilde{\mathbb{1}}^{A_{\mathrm{O}}} = \mathbb{1}^{A_{\mathrm{O}}}/d_{A_{\mathrm{O}}}$ and $d_{A_{\mathrm{O}}}$ is the dimension of the system A_{O}. When this condition is satisfied, the party A cannot signal to any party and is considered *last*. In the case where the output system of the party is decomposed into output subsystems A_{O_i}, $i = 1, \ldots, n$, then the corresponding identity matrix in the process matrix lives on the Hilbert space of that output subsystem A_{O_i}. We also call this subsystem *open* and the linear constraint is

$$\widetilde{\mathbb{1}}^{A_{\mathrm{O}_i}} \otimes \mathrm{Tr}_{A_{\mathrm{O}_i}} W = W. \tag{4}$$

Channel: A quantum channel between the input of a party A and its parents space Γ^A is represented by a factor $T^{\Gamma^A A_{\mathrm{I}}}$ in the process matrix, as we have already mentioned. It is a positive matrix that lives on the tensor product of the Hilbert spaces of the output and input systems involved, and has the property that upon tracing out the output of the channel (the input of A) what remains is identity on the input (the space of output systems Γ^A):

$$\mathrm{Tr}_{A_{\mathrm{I}}} T^{\Gamma^A A_{\mathrm{I}}} = \mathbb{1}^{\Gamma^A}. \tag{5}$$

This property is necessary and sufficient for the channel to be trace preserving and we use it to discover channels in the process matrix: we trace out the input of A, A_{I}, and we check whether in the remaining process matrix there is identity on the output system of a given party, say, B. This describes a linear constraint that a process matrix satisfies when there is a channel from the output of B to the input of A.

$$\widetilde{\mathbb{1}}^{B_{\mathrm{O}}} \otimes \mathrm{Tr}_{B_{\mathrm{O}}}(\mathrm{Tr}_{A_{\mathrm{I}}} W) = \mathrm{Tr}_{A_{\mathrm{I}}} W. \tag{6}$$

If the output of party B is decomposed into subsystems, then we use the above constraint for each subsystem separately, by replacing B_{O} with every output subsystem B_{O_i}.

$$\widetilde{\mathbb{1}}^{B_{\mathrm{O}_i}} \otimes \mathrm{Tr}_{B_{\mathrm{O}_i}}(\mathrm{Tr}_{A_{\mathrm{I}}} W) = \mathrm{Tr}_{A_{\mathrm{I}}} W. \tag{7}$$

Note that conditions (6) and (7) are also satisfied for open systems and subsystems, respectively. However, the algorithm checks conditions (3) and (4) first and does not consider again those (sub)system that have been tagged "open". Therefore, it will not associate a channel to open (sub)systems. The maximal set of output systems and subsystems for which conditions (6) and (7) hold is the parent space of A, Γ^A.

In the concrete implementation of the algorithm, the above equalities are tested up to some precision defined by a small number ε, which can be adjusted depending on the working precision. When testing examples generated numerically, this permits one to take into account the different numerical rounding of non-integer numbers that might otherwise lead to errors—for example, $\sqrt{2}$ defined up to some digit will be different to $\sqrt{2^2}/\sqrt{2}$ as the rounding of the last digit in different steps of the calculation will cause a different result. Naturally, the number ε can also be adjusted to account for experimental inaccuracies, when the process matrix is obtained from experimental data.

The code. The causal discovery code subjects the process matrix to the above types of linear constraints and the set of them that are satisfied defines the DAG.

The code takes as input: the number of parties, the dimension of each input system, output system, output subsystem, and the process matrix. The code assumes that the process matrix is positive semi-definite. Hence, its output is meaningful only if this assumption is satisfied.

Briefly the procedure of causal discovery goes as follows: First the code identifies and traces out any open output subsystems. Then it determines whether the process matrix is causally ordered. If it is, it outputs a possible causal order and proceeds to determine if the process is Markovian. For a Markovian process, it outputs the DAG, and the represented mechanisms. Below we expand on these three stages. In the Methods, we show how the code works using an example of a 4-partite process matrix. We present the causal information extracted in the different stages for that example, as well as the final output of the code.

Tracing out open output subsystems: The code checks each output subsystem to identify if it is open, using the linear constraint in Eq. (4). Each found open subsystem is traced out from the process matrix, keeping track of the label of the party and the label of the subsystem, for example, subsystem 3 of party 2. Keeping track of open subsystems is what allows the algorithm to find a *minimal* DAG, namely without extra arrows, as discussed below.

Checking if W *is causally ordered*: Let us call a *non-signaling set*, a set of parties that are causally independent, namely that cannot signal to each other. A non-signaling set is *maximal* if it is not a proper subset of another non-signaling set. The first output of the algorithm is all the maximal non-signaling sets and their causal order. This is done through the linear constraint that detects open output systems, in Eq. (3). The set of parties whose output systems satisfy the constraint is labeled as *last set*. Note that the constraints has to be satisfied by the whole output system and not only by some subsystems. To determine the next set, the *second last*, the code traces out the last set from the process matrix and, using the same constraint, it identifies the new last set, and so on. Note that the partition into maximal non-signaling sets does not uniquely identify the partial order of the parties, in the sense that it is not guaranteed that parties in different non-signaling sets can signal to each other. What is guaranteed is that at least one party from a set $\mathcal{X}$ can signal to at least one party in a succeeding set $\mathcal{Y}$ (Fig. 4). Note also that the partition into maximal non-signaling sets is not unique, much like a foliation of space–time into space-like hypersurfaces.

The process matrix is causally ordered if and only if the algorithm succeeds in grouping all parties in maximal non-signaling sets. This is because, given the non-signaling sets, we can define a total order among the parties by adding arbitrary order relations among members of each set. For example, we can order the parties in different time steps where: when $A \prec B$, A occurs at a time before B and, when $A \| B$ then we pick an arbitrary time ordering (Fig. 5). With the parties ordered in this way, the process matrix satisfies the condition defining a quantum comb.[20] This is a recursive version of Eq. (3), that holds for the output of each system after all systems that come after it are traced out. A central result in the theory of quantum networks is that, whenever this condition holds, the corresponding process can be realized as a channel with memory.[20,26,27] Thus, this part of the algorithm determines whether the input process matrix has a physical realisation as a causally ordered process.

Causal discovery and Markovianity: After the algorithm has traced out all open output subsystems and has established the maximal non-signaling sets of the parties, it is time to determine the DAG. The algorithm checks all possible causal arrows—compatible with the previously found causal order—between pairs of an input system of a party and an output system of another party, using Eq. (6). If a party's output system is divided into subsystems, then each subsystem is checked using the linear constraint in Eq. (7). To check if these constraints are satisfied, the algorithm has to check each possible one individually. In particular, for each input system that is traced out, it checks whether the constraint holds for each output system or subsystem that has not been associated yet with a causal arrow. Every time the constraint is satisfied, a causal arrow is associated with the corresponding systems and the output system or subsystem is marked as used and is not being checked again. The collection of all output systems and subsystems that satisfy the constraints for a single input system of a party A uniquely identifies the parent space of A, Γ^A. Figure 6 shows the information input to the code, and the output information that is obtained during the three stages described above.

At this stage, the code outputs a DAG if the process is Markovian, namely if the process matrix is of the form of Eq. (2). To determine this, the code constructs a test-matrix that is Markovian with respect to the found DAG: it contains all (and only) the factors as in Eq. (2) that correspond to the elements of the DAG. There are three kinds of these elements: first parties, causal arrows, and last parties; the corresponding terms on the process matrix are input states for the first parties, channels that live on the input and output systems and subsystems of the associated parties, and identity matrices on the output system of the last parties, respectively. To construct the test process matrix, these factors are extracted from the original process matrix by tracing out all systems except from the desired ones. If the process is Markovian,

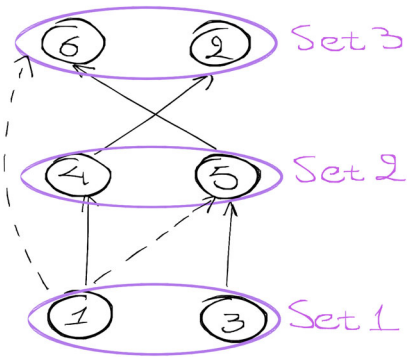

Fig. 4 Maximal non-signaling sets. The first output of the code is a grouping of the parties into maximal non-signaling sets. The solid arrows represent a DAG compatible with this grouping. Not all parties in different sets are linked by causal arrows; the dashed arrows are examples of these missing links. Credits: C. Giarmatzi

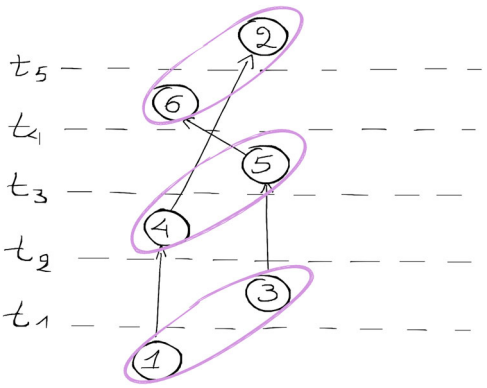

Fig. 5 Total causal order. Starting from the DAG in Fig. 4, we can order all events in time, obtaining a total order of the parties, by putting an arbitrary order between parties in the same non-signaling set. Credits: C. Giarmatzi

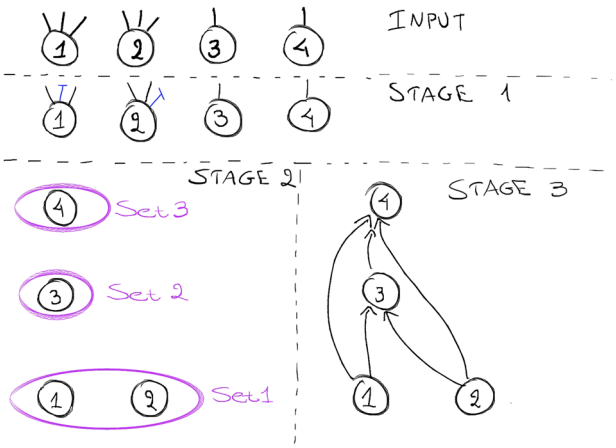

Fig. 6 Stages of the algorithm. As part of the input, we depict the parties and the information about their output systems and subsystems. Stage one traces out the open output subsystems, depicted in blue. The rest of the systems in black are output systems and subsystems. For a causally ordered process, stage two groups the parties into maximal non-signaling sets. For a Markovian process, stage three provides the causal model. There is no arrow for the output system of party 4 as it is last. Credits: C. Giarmatzi

then the test-matrix will be equal to the original process matrix that was input to the code.

Minimality. The code is guaranteed to give a unique and minimal DAG for a Markovian process. A process matrix is said to be Markov with respect to the DAG if every channel (found by Eqs. (6) and (7)) in the process matrix is represented by an arrow in the DAG. However, a W can be Markov to more than one DAG—some DAGs will have arrows allowed by the causal order but there is no actual channel in the W corresponding to this arrow. In other words, a W can be in the tensor product form (2), but with some factor of the form $T^{\Gamma^M M_i} = \mathbb{1}^{\Gamma^M} \otimes \rho^{M_i}$, for some normalized density matrix ρ. This represents a channel that always produces the state ρ. Hence, this W is Markovian with respect to a DAG with arrows representing such channels, from Γ^M to M_i, but is also Markovian to a DAG without such arrows.

If every arrow in the DAG corresponds to a non-trivial channel in the process matrix, the DAG is called *minimal*. From another perspective, a DAG is minimal if, by removing any arrow from it, then the W is not any more Markov with respect to the resulting DAG.

The fact that the output of the code is always the minimal DAG is guaranteed by the first step of the algorithm, where the open subsystems are established and discarded. Indeed, an "extra arrow" in a non-minimal DAG would necessarily be associated with an open subsystem—an identity tensor factor in the process matrix.

Note also that, in ref. [15] it was proven that a DAG can be in principle recovered under the additional assumption of *faithfulness*. Our algorithm does not require such an extra assumption, proving that causal discovery is always possible for a quantum Markov causal model.

Complexity of the algorithm

The dimension of the process matrix is given by the product of input and output dimensions of each party. Thus, the size of the process matrix would generally scale exponentially with the number of parties. This is expected, as also the dimension of ordinary density matrices would scale exponentially with the number of parties.

One can however consider situations where, under appropriate assumptions and approximations, the physical scenario under consideration is described by a polynomial number of parameters. Then, the main cost of the algorithm lies in two parts: the one that establishes the non-signaling sets and the one that searches for causal arrows between parties. The first step tests condition (3) for all parties, to determine each non-signaling set, and the second step tests condition (6) (or (7)) for pairs of nodes—in both cases the number of tests required is thus quadratic in the number of parties. Therefore, given an efficient encoding of the input process matrix, the algorithm scales quadratically with the number of parties.

Non-Markovian processes

A Markovian process is one with a process matrix of the form of Eq. (2), and is represented by a DAG. In a non-Markovian process the process matrix is not of that form, i.e. it is not a tensor product of factors representing input states for the parties with no incoming arrows, channels, and identity matrices for the output of the parties in the last set. In other words, in a non-Markovian process, these factors alone—or their representation in a DAG—cannot account for the observed correlations between the events.

Latent nodes. If the code outputs that the process is causally ordered but non-Markovian, it can be represented as a quantum circuit compatible with the causal order, where the parties are connected with quantum channels with memory,[20,27] as we depict in Fig. 7(a). In the language of causal modelling, such a process can be represented by an extended DAG with additional nodes, called *latent*, and channels connecting them to the rest of the parties, so that the extended process is Markovian and reduces to the original one for a particular choice of CPTP maps applied in the extra nodes,[15] as depicted in Fig. 7(b). The intuition is that the correlations obtained from the original process cannot be produced by considering the original nodes and channels without memory. Therefore, there are extra nodes, not considered in the process, which affect the local outcomes of the nodes considered.

For example, the outcomes of quantum measurements performed in some measurement stations (nodes) in a laboratory, may be affected by the temperature or maybe another system is leaking into one of the stations, like stray light affecting the detection part and causing correlated noise. If these are producing significant change in the data—higher than the noise tolerance in the code—the process will appear non-Markovian.

To recover a causal model by introducing latent nodes we would need to extend the algorithm such that it adds nodes and arrows until it finds that it is Markovian. Computationally, this task can be hard because the code has to find the right combination of the number of nodes needed, their position in the DAG and the exact channels around them. However, although the original process is non-Markovian, the code still outputs the causal order of the parties for a causally ordered process matrix. From that, one could make guesses about the right causal model, by introducing nodes with specific input and output systems and channels connecting them to the rest of the parties. To do this, one should add the corresponding factors into the current test matrix W_{test} and run the code using as input the updated number of parties, dimensions of systems and W_{test} as the process matrix and see if now the process is Markovian.

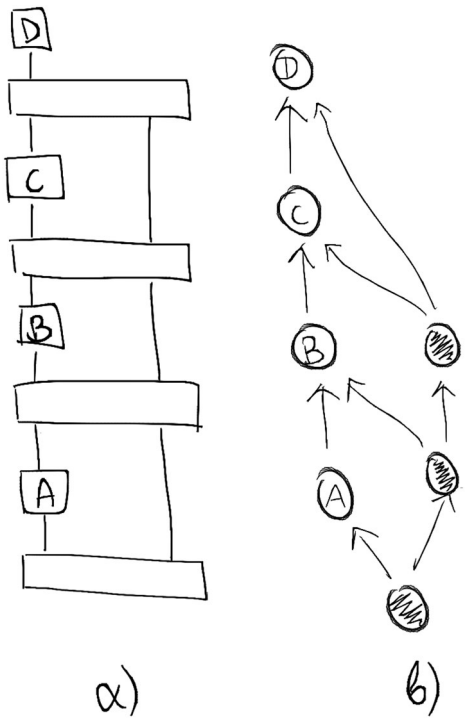

Fig. 7 Non-Markovian vs. Markovian process. In figure (**a**), we represent a causally ordered non-Markovian process as a quantum circuit where the channels connecting the parties are quantum channels with memory. In figure (**b**), the same process can be represented as a Markovian process for the extended number of nodes. The new nodes introduced are the latent nodes. Credits: C. Giarmatzi

Mixture of causal orders. Another possible reason why the process is non-Markovian is that it might be the case that it represents a probabilistic mixture of two or more Markovian processes with different causal orders, resulting in a non-causally ordered process matrix. There is a Semidefinite Program (SDP) for this problem, that finds the right decomposition.[28] For instance, for a bipartite process, the SDP would look like the following.

$$\begin{aligned} &\text{given } W \\ &\text{find } q \\ &\text{such that } W = qW^{A\prec B} + (1-q)W^{B\prec A} \\ &0 \le q \le 1 \end{aligned} \tag{8}$$

where $W^{X\prec Y}$ denotes a valid process matrix where Y is last and therefore has a factor $\mathbb{1}^{Y_O}$. In the case with more parties, one simply has to write a decomposition that includes all different causal orders for the given parties. Given the result, one can apply the causal discovery algorithm to each term in the decomposition. Note that a mixture of processes with the same causal order can be modeled as a causally ordered, non-Markovian process with latent nodes acting as "classical common causes".[29,30]

Dynamical and indefinite causal order. So far, we have seen that when events have a definite causal order, they can be represented either by a fixed causal order process or by a mixture of causal orders. However, it may be the case that the process matrix represents a situation of more than two parties, where the causal order of some parties depend on the operations of parties in their past. That is, a party may influence the causal order of future parties. Such a dynamical causal order was studied in ref. [31] where a definition of causality was proposed, compatible with such dynamical causal order. For the tripartite case, it was found that the process matrix describing such a situation should obey certain conditions. However, similar conditions were not found for the case of arbitrary parties. In such cases, the notion of causal discovery is not clear, as depending on some events in the past, the DAG of future ones would change. Hence the output would be different DAGs for different operations of certain parties. We do not know if the discovery of those DAGs is possible.

Different definitions of Markovianity. Our algorithm relies on the definition of quantum Markov causal model of ref. [15]. A different definition was proposed in ref. [16], where the output systems of the parties are not assumed to factorize into subsystems in the presence of multiple outgoing arrows. In ref. [16], arrows in the DAG are still associated with a quantum channel from the output space of the parent nodes to the input space of the child but, rather than defining a factorisation in subsystems of the output space, multiple outgoing arrows are more generally associated with commuting channels. For example, in a tripartite scenario where A is a parent of both B and C, a Markovian process matrix would have the form

$$W^{A_I A_O B_I B_O C_I C_O} = \rho^{A_I} \otimes \left(T_1^{A_O B_I} \cdot T_2^{A_O C_I}\right) \otimes \mathbb{1}^{B_O C_O}, \tag{9}$$

with the condition $T_1^{A_O B_I} \cdot T_2^{A_O C_I} = T_2^{A_O C_I} \cdot T_1^{A_O B_I}$. Thus, according to ref. [16], a Markovian process matrix does not need to be a tensor product but can more generally be a product of commuting matrices. To distinguish the two definitions, we will call *tensor-Markovian* and *commuting-Markovian* a process matrix that satisfies the condition of ref. [15] (used in our code) and ref. [16], respectively. Note that all tensor-Markovian processes are commuting Markovian, but the converse is not true. In ref. [16] it is further assumed that input and output spaces of each node are isomorphic. Thus, strictly speaking, not all tensor-Markovian process considered here satisfy the definition of ref. [16], but only those with input and output of equal dimension. This difference is of little consequence from the point of view of a causal discovery algorithm, since in any case the dimension of each space has to be specified as input to the code.

Our algorithm could be adapted to discover the causal structure of commuting-Markovian processes. Note that the strategy used in our code, to detect the parent space of each node by checking (5), would not work. Indeed, tracing out B_I from matrix (9) does not result in a matrix with identity on A_O. A possible approach could be to instead detect all the *children* of each node A, namely all the nodes with an incoming arrow departing from A. The children are then identified as the smallest subset of parties $C^1, \ldots, C^k$ such that

$$\widetilde{\mathbb{1}}^{A_O} \otimes \mathrm{Tr}_{A_O}\left(\mathrm{Tr}_{C_I^1,\ldots,C_I^k} W\right) = \mathrm{Tr}_{C_I^1,\ldots,C_I^k} W. \tag{10}$$

As this condition must be checked for subsets of parties, the number of tests is exponential in the number of parties for the worst-case scenario. In contrast, we have seen that to discover a tensor-Markovian causal structure a quadratic number of tests is sufficient. Another potential complication is that our test for Markovianity relies on the tensor-product form of the process matrices; it is not clear if there is a simple way to test whether a process is commuting-Markovian.

An alternative approach is to retain the definition of tensor-Markovian processes and model commuting-Markovian processes as non-Markovian ones. Indeed, since a commuting-Markovian process is causally ordered, it can always be recovered from a tensor-Markovian one by adding an appropriate number of latent nodes.[15] An extension of our code to detect latent nodes could thus be used to detect the causal structure of a commuting-Markovian process. In Fig. 8 we show an example of a DAG of a commuting-Markovian process (left) and how that would be represented as a tensor-Markovian (right) with a latent node.

DISCUSSION

We have presented an algorithm (whose implementation we provide) that can discover an initially unknown causal structure in a quantum network. This is an important proof of principle: it shows that causal structure has a precise empirical meaning in quantum mechanics. Just as other physical properties, it can be unknown and discovered. This is of particular significance for foundational approaches where causal structure is seen as emergent from more fundamental primitives. Causal discovery provides the methodology to determine when and how causal structure emerges.

Causal discovery can also have broad applications for protocols based on large and complex quantum networks. Our algorithm is guaranteed to find a minimal causal model for any Markovian process, namely a process in which all causally relevant events are under experimental control, with no extra assumptions; this improves on the results of ref. [15], where the additional condition of faithfulness was invoked. Even for non-Markovian processes,

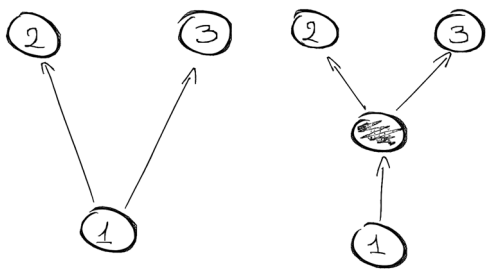

Fig. 8 Different definitions of Markovianity. A process that is Markovian according to ref. [16], e.g. for the DAG on the left, is generally described by a DAG with a latent node (filled node in the DAG on the right) according to the definition of Markovianity of ref. [15] on which our algorithm is based. Credits: C. Giarmatzi

the algorithm still recovers important causal information, namely a causal order of the events.

Another important use of our algorithm is to tackle the difficult problem of non-Markovianity. An extensive body of research is currently devoted to the problem of detecting non-Markovianity.[32] Our algorithm finds a concrete solution: it allows discovering when some external memory is affecting the correlations in the observed system. Detecting non-Markovianity can also have important practical applications for large quantum networks: the presence of "latent nodes", can indicate a possible source of systematic correlated noise in a process, that might affect the working of a quantum protocol. It can further have applications in cryptography for detecting the presence of an eavesdropper.

Finally, our algorithm has promising possible extensions. A natural extension is an algorithm that can make "good guesses" for causal structure in the presence of latent nodes. Promising is also the extension of causal discovery to mixtures of causal order, dynamical, and indefinite causal structure.

METHODS

In this section, we provide an example of how the code works for a particular process matrix, and how the different levels of causal information are extracted. In our example we have four parties {1, 2, 3, 4}, with dimensions

$$\mathrm{dim} = \begin{bmatrix} d_{1_\mathrm{I}} & d_{1_\mathrm{O}} \\ d_{2_\mathrm{I}} & d_{2_\mathrm{O}} \\ d_{1_\mathrm{I}} & d_{1_\mathrm{O}} \\ d_{2_\mathrm{I}} & d_{2_\mathrm{O}} \end{bmatrix} = \begin{bmatrix} 2 & 4 \\ 2 & 8 \\ 2 & 2 \\ 2 & 4 \end{bmatrix}.$$

Party 1 has two output subsystems with a dimension of 2 each, and party 2 has three output subsystems with a dimension of 2 each, denoted as subdim{1} = [2 2], subdim{2} = [2 2 2]. The process matrix is of the following form:

$$W_{\mathrm{input}}^{1_\mathrm{I}1_\mathrm{O}2_\mathrm{I}2_\mathrm{O}3_\mathrm{I}3_\mathrm{O}4_\mathrm{I}4_\mathrm{O}} = \rho^{3_\mathrm{I}} \otimes T^{3_\mathrm{O}1_\mathrm{I}} \otimes T^{1_{\mathrm{O}_1}2_\mathrm{I}} \otimes T^{2_{\mathrm{O}_3}1_{\mathrm{O}_2}4_\mathrm{I}} \otimes \mathbb{1}^{2_{\mathrm{O}_1}2_{\mathrm{O}_2}4_\mathrm{O}}, \tag{11}$$

where ρ is some input state for party 3, and 1_{O_i} and 2_{O_i} denote the ith output subsystem of party 1 and 2, respectively. Note that the above form of the input process matrix to the code is of course not known in advance. We remind that the input to the code is the above matrix dim, the arrays subdim{1}, subdim{2} and the process matrix W_{input}, in its numerical form, in which the systems are ordered as $1_\mathrm{I}1_{\mathrm{O}_1}1_{\mathrm{O}_2}2_\mathrm{I}2_{\mathrm{O}_1}2_{\mathrm{O}_2}2_{\mathrm{O}_3}3_\mathrm{I}3_\mathrm{O}4_\mathrm{I}4_\mathrm{O}$. In the following, we describe the calculations that take place. The various procedures can be grouped into three stages.

Stage 1—tracing out the open output subsystems

In this stage, the code looks at the elements subdim{X}. In our example, $X = 1, 2$. Knowing that these parties have output subsystems, it checks if those are open—if on the process matrix there is identity on those subsystems. To do that, the code checks the following equality for each

```
>> causal_discovery_algorithm

the_sets =

     4
     2
     1
     3

Time 8.1967

There are open subsystems: 1 of party 2 of dimension 2  --- *** ---
There are open subsystems: 2 of party 2 of dimension 2  --- *** ---
Link from subsystem 3 of party 2 to party 4.  --- *** ---
Link from subsystem 1 of party 1 to party 2.  --- *** ---
Link from party 3 to party 1.  --- *** ---

3 primal arrows

primal_arrows =

     2     4
     1     2
     3     1

Time 2.0232

Link from subsystem 2 of party 1 to party 4.  --- *** ---

1 secondary arrows

secondary_arrows =

     1     4

Time 0.089431

The process is Markovian
>>
```

Fig. 9 Output of command window. The command window, for the given example, showing the output of the code regarding the maximal non-signaling sets, the open subsystems and the causal arrows. Primal_arrows refers the the causal arrows from successive maximal non-signaling sets and secondary_arrows refers to all the other causal arrows. Time refers to the time that lapsed to evaluate the step just above

output subsystem:

$$\widetilde{\mathbb{1}}^{A_{O_i}} \otimes \mathrm{Tr}_{A_{O_i}} W_{\mathrm{input}} = W_{\mathrm{input}}. \tag{12}$$

The code displays on the command window (see Fig. 9) the output subsystems for which this constraint is satisfied, and traces it out from the process matrix. In our example, it outputs "There are open subsystems: 1 of party 2 of dimension 2" and "2 of party 2 of dimension 2". The remaining process matrix is now

$$W^{1_I 1_O 2_I 2_O 3_I 3_O 4_I 4_O} = \rho^{3_I} \otimes T^{3_O 1_I} \otimes T^{1_{O_1} 2_I} \otimes T^{2_{O_3} 1_{O_2} 4_I} \otimes \mathbb{1}^{4_O}, \tag{13}$$

and will be used as the input process matrix for the rest of the code.

Stage 2—checking if W is causally ordered

In this stage, the maximal non-signaling sets are established, as well as their causal order. To establish the "last set", which is the set of parties that have no outgoing arrow, the code checks the constraint

$$\widetilde{\mathbb{1}}^{A_O} \otimes \mathrm{Tr}_{A_O} W = W, \tag{14}$$

for all parties $A = \{1, 2, 3, 4\}$. The set of parties that satisfy this constraint constitutes the "last set". To establish the next set, the last set is traced out from the process matrix and the remaining process matrix undergoes the same above constraint for the remaining parties. In this way, all the maximal sets are established, together with their causal order. If the code completes this task with all the parties grouped into maximal sets, then the process matrix is causally ordered. In our example the maximal sets and their causal order are: $\{3\} \prec \{1\} \prec \{2\} \prec \{4\}$. This is shown on the command window as (see also Fig. 9)

$$\text{the\_sets} = \begin{bmatrix} 4 \\ 2 \\ 1 \\ 3 \end{bmatrix}.$$

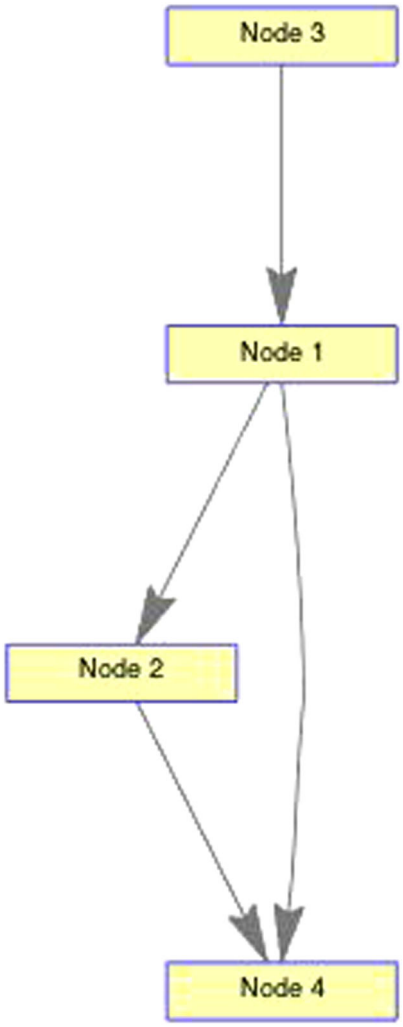

Fig. 10 Output DAG. The DAG that the code outputs for the given example

Stage 3—causal discovery and Markovianity

In this stage the code discovers the causal arrows that connect the parties. Once all the causal arrows have been found, it checks if the process is Markovian. If it is, it outputs the DAG corresponding to the process. If it is not Markovian, then the discovered causal arrows are not reliable and hence a DAG is not provided. Now let us see how the code goes about discovering the causal arrows. The causal arrows are between an input system of a party, say A and an output system or subsystem of another party, say B. This is done by the following two constraints for output system or subsystem.

$$\widetilde{\mathbb{1}}^{B_O} \otimes \mathrm{Tr}_{B_O}(\mathrm{Tr}_{A_I} W) = \mathrm{Tr}_{A_I} W, \tag{15}$$

$$\widetilde{\mathbb{1}}^{B_{O_i}} \otimes \mathrm{Tr}_{B_{O_i}}(\mathrm{Tr}_{A_I} W) = \mathrm{Tr}_{A_I} W. \tag{16}$$

To check whether this constraint is satisfied, the code must check each pair of [input system–output system] or [input system–output subsystem] individually. An input system can be involved with more than one causal arrow, but an output system or subsystem can be involved with only one causal arrow. Hence, once an output system or subsystem has been associated with a causal arrow, it is not checked again in the rest of the code. The code outputs on the command window the causal arrows found. In our example that would be "Link from subsystem 3 of party 2 to party 4.", "Link from subsystem 1 of party 1 to party 2.", "Link from party 3 to party 1.", "Link from subsystem 2 of party 1 to party 4.", as is shown in Fig. 9.

Now the code proceeds with the Markovianity check. This involves constructing a process matrix that is Markovian with respect to the found DAG; specifically, a matrix composed out of input states for the first parties, channels for the found causal arrows and identity matrices for the last parties. For the first parties, it extracts from the process matrix their input states. In our example, party 3 is first and its input state is extracted from the process matrix by tracing out all the other systems

$$\rho^{3_I} = \mathrm{Tr}_{\widetilde{3_I}} W, \tag{17}$$

where $\widetilde{3_I}$ denotes the space of input and output systems that is complementary to 3_I. To extract the channels, the code similarly traces out all systems except the ones involved in the channels. Note that for an input system that is involved with many arrows the corresponding channel would be represented from the parent space of the party—all the systems that have an arrow to that party—to the input of the party. In our example, there are the simple channels from systems 3_O to 1_I, from 1_{O_1} to 2_I, and from $\{1_{O_2}, 2_{O_3}\}$ to 4_I. The corresponding channels are

$$T^{3_O 1_I} = \mathrm{Tr}_{\widetilde{3_O 1_I}} W \tag{18}$$

$$T^{1_{O_1} 2_I} = \mathrm{Tr}_{\widetilde{1_{O_1} 2_I}} W \tag{19}$$

$$T^{I_{O_2} 2_{O_3} 4_I} = \mathrm{Tr}_{\widetilde{I_{O_2} 2_{O_3} 4_I}} W \tag{20}$$

The code is then adding identities to the output systems of the last parties. In our example we have 1^{4_O}. Finally, the code constructs the following test matrix:

$$W^{1_I 1_O 2_I 2_O 3_I 3_O 4_I 4_O}_{\mathrm{test}} = \rho^{3_I} \otimes T^{3_O 1_I} \otimes T^{1_{O_1} 2_I} \otimes T^{2_{O_3} 1_{O_2} 4_I} \otimes \mathbb{1}^{4_O}. \tag{21}$$

After rearranging the systems in the order of the original process matrix, that is $[1_I 1_O 2_I 2_O 3_I 3_O 4_I 4_O]$, the code tests if $W_{\mathrm{test}} = W$. If this is true, which it is in our example, the code outputs on the command window "the process is Markovian" and outputs the DAG corresponding to the found causal arrows, shown in Fig. 10.

Code availability

In the given repository (https://github.com/Christina-Giar/quantum-causal-discovery-algo.git), we provide the code presented in this paper, written in MatLab, together with the set of necessary functions. We also provide a

code written in Mathematica, where valid process matrices of arbitrary causal structures can be generated, given the number of parties. These process matrices can be used as examples of input to the code. Finally, we provide a Manual on how to use both codes.

ACKNOWLEDGEMENTS

We thank Gerard Milburn, Sally Shrapnel and Andrew White for discussions. We acknowledge the traditional owners of the land on which the University of Queensland is situated, the Turrbal and Jagera people. This work was supported by the Australian Research Council (ARC) Centre for Engineered Quantum Systems grant (CE 110001013), the ARC Centre for Quantum Computation and Communication Technology (Grant No. CE110001027), and by the Templeton World Charity Foundation (TWCF 0064/AB38). This publication was made possible through the support of a grant from the John Templeton Foundation. The opinions expressed in this publication are those of the authors and do not necessarily reflect the views of the John Templeton Foundation.

AUTHOR CONTRIBUTIONS

Both authors developed the theory and contributed to writing of the manuscript. C. Giarmatzi implemented the MatLab code.

ADDITIONAL INFORMATION

Competing interests: The authors declare no competing financial interests.

Publisher's note: Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

REFERENCES

1. Pearl, J. *Causality* (Cambridge University Press, Cambridge, 2009).
2. Spirtes, P., Glymour, C. N. & Scheines, R. *Causation, prediction, and search*, Vol. 81 (MIT Press, Cambridge, MA, 2000).
3. Lamport, L. Time, clocks, and the ordering of events in a distributed system. *Commun. ACM* **21**, 558–565 (1978).
4. Wood, C. J. & Spekkens, R. W. The lesson of causal discovery algorithms for quantum correlations: causal explanations of Bell-inequality violations require fine-tuning. *New J. Phys.* **17**, 033002 (2015).
5. Tucci, R. R. Quantum Bayesian nets. *Int. J. Mod. Phys. B* **09**, 295–337 (1995).
6. Leifer, M. S. Quantum dynamics as an analog of conditional probability. *Phys. Rev. A* **74**, 042310 (2006).
7. Laskey, K. B. Quantum causal networks. Preprint at arXiv:0710.1200 [quant-ph] (2007).
8. Leifer, M. S. & Spekkens, R. W. Towards a formulation of quantum theory as a causally neutral theory of bayesian inference. *Phys. Rev. A* **88**, 052130 (2013).
9. Cavalcanti, E. G. & Lal, R. On modifications of reichenbach's principle of common cause in light of bell's theorem. *J. Phys. A* **47**, 424018 (2014).
10. Fritz, T. Beyond Bell's theorem II: scenarios with arbitrary causal structure. *Commun. Math. Phys.* **341**, 391–434 (2016).
11. Henson, J., Lal, R. & Pusey, M. F. Theory-independent limits on correlations from generalized bayesian networks. *New J. Phys.* **16**, 113043 (2014).
12. Pienaar, J. & Brukner, Č. A graph-separation theorem for quantum causal models. *New J. Phys.* **17**, 073020 (2015).
13. Chaves, R., Majenz, C. & Gross, D. Information-theoretic implications of quantum causal structures. *Nat. Commun.* **6,** https://doi.org/10.1038/ncomms6766 (2015).
14. Ried, K. et al. A quantum advantage for inferring causal structure. *Nat. Phys.* **11**, 414–420 (2015).
15. Costa, F. & Shrapnel, S. Quantum causal modelling. *New J. Phys.* **18**, 063032 (2016).
16. Allen, J.-M. A., Barrett, J., Horsman, D. C., Lee, C. M. & Spekkens, R. W. Quantum common causes and quantum causal models. *Phys. Rev. X* **7**, 031021 (2017).
17. Shrapnel, S. Discovering quantum causal models (2015).
18. Shrapnel, S. *Using Interventions to Discover Quantum Causal Structure*. Ph.D. thesis (2016).
19. Oreshkov, O., Costa, F. & Brukner, Č. Quantum correlations with no causal order. *Nat. Commun.* **3**, 1092 (2012).
20. Chiribella, G., D'Ariano, G. M. & Perinotti, P. Theoretical framework for quantum networks. *Phys. Rev. A* **80**, 022339 (2009).
21. Jamiołkowski, A. Linear transformations which preserve trace and positive semidefiniteness of operators. *Rep. Math. Phys.* **3**, 275–278 (1972).
22. Choi, M.-D. Completely positive linear maps on complex matrices. *Linear Algebra Appl.* **10**, 285–290 (1975).
23. Caves, C. M., Fuchs, C. A., Manne, K. K. & Renes, J. M. Gleason-type derivations of the quantum probability rule for generalized measurements. *Found. Phys.* **34**, 193–209 (2004).
24. Shrapnel, S., Costa, F. & Milburn, G. Updating the Born rule (2017). Preprint at arXiv:1702.01845 [quant-ph].
25. Pollock, F. A., Rodríguez-Rosario, C., Frauenheim, T., Paternostro, M. & Modi, K. Complete framework for efficient characterisation of non-Markovian processes. Preprint at arXiv:1512.00589 [quant-ph] (2015).
26. Gutoski, G. & Watrous, J. Toward a general theory of quantum games. In *Proceedings of 39th ACM STOC*, 565–574. Preprint at arXiv:quant-ph/0611234 (2006).
27. Kretschmann, D. & Werner, R. F. Quantum channels with memory. *Phys. Rev. A* **72**, 062323 (2005).
28. Araújo, M. et al. Witnessing causal nonseparability. *New J. Phys.* **17**, 102001 (2015).
29. MacLean, J.- P. W., Ried, K., Spekkens, R. W. & Resch, K. J. Quantum-coherent mixtures of causal relations. Preprint at arXiv:1606.04523 [quant-ph] (2016).
30. Feix, A. & Brukner, Č. Quantum superpositions of "common-cause" and "direct-cause" causal structures. Preprint at arXiv:1606.09241 [quant-ph] (2016).
31. Oreshkov, O. & Giarmatzi, C. Causal and causally separable processes. *New J. Phys.* **18**, 093020 (2016).
32. Rivas, Á., Huelga, S. F. & Plenio, M. B. Quantum non-Markovianity: characterization, quantification and detection. *Rep. Prog. Phys.* **77**, 094001 (2014).