Available online at www.sciencedirect.com
ScienceDirect
International Journal of Approximate Reasoning
46 (2007) 274–299
INTERNATIONAL JOURNAL OF APPROXIMATE REASONING
www.elsevier.com/locate/ijar

Inference in multi-agent causal models

Sam Maes *, Stijn Meganck, Bernard Manderick

Computational Modeling Lab, Vrije Universiteit Brussel, Pleinlaan 2, 1050 Brussel, Belgium

Received 3 July 2005; received in revised form 15 March 2006; accepted 21 September 2006
Available online 15 November 2006

Abstract

In this article, we demonstrate the usefulness of causal Bayesian networks as probabilistic reasoning systems. The biggest advantage of causal Bayesian networks over traditional probabilistic Bayesian networks is that they sometimes allow to perform causal inference, i.e. the calculation of the causal effect of one variable on other variables. We treat a state-of-the-art algorithm for performing causal inference that is based on a new factorization of the joint probability distribution and is a systematic approach for the calculation due to Tian and Pearl.

We elaborate on the problems that can arise when working with a centralized approach and discuss how a decentralized cooperative multi-agent approach might overcome some of these problems.

The main contribution of this article is the introduction of multi-agent causal models as a way to overcome the problems in a centralized setting. They are an extension of causal Bayesian networks to a distributed setting consisting of a number of agents each having access to an overlapping set of the variables. We extend a state-of-the-art causal inference algorithm for this particular domain. We will show that our approach is as powerful in computing causal effects as the centralized algorithm.

Keywords: Causal models; Multi-agent systems; Causal inference

1. Introduction

This article is situated in the area of causal Bayesian networks. In their basic form, the qualitative part of these models consists of a directed acyclic graph, where the nodes are

* Corresponding author.
E-mail addresses: sammaes@vub.ac.be (S. Maes), smeganck@vub.ac.be (S. Meganck), bmanderi@vub.ac.be (B. Manderick).

0888-613X/$ - see front matter © 2006 Elsevier Inc. All rights reserved.
doi:10.1016/j.ijar.2006.09.005

variables and the edges connecting them represent causal relations between the corresponding variables. The causal relation between two variables, in the form of a directed edge from a cause variable C to an effect variable E, is understood as the effect a manipulation of variable C (the cause) would have on variable E (the effect). More specifically, when one would vary C by performing a surgical manipulation on variable C, i.e. a manipulation that only changes C all else staying equal (ceteris paribus), then variable E would also vary. Typically, the quantitative part of a causal graphical model consists of a set of conditional probability distributions, of each variable conditional on its parent variables in the graph.

These causal models can be obtained in several ways, ranging from randomized experiments to expert knowledge and learning algorithms and combinations of all the previous.

Next to the intuitively appealing form of causal graphical models, their main advantage is that they can help performing causal inference or identification. This is the process of calculating the effect a manipulation of a variable would have on other variables in the model, without actually performing that manipulation. This is particularly useful in tools for decision support, where the utility of a possible policy choice can be tested before actually performing that action. When all relevant variables in a domain are observed, the process of causal inference is rather trivial, but when some variables are unobserved, the problem becomes much more difficult and in some cases impossible.[1] Two example approaches to causal inference in causal Bayesian networks are *do-calculus* [1] and the state-of-the-art algorithm based on a new factorization of the joint probability distribution of all the variables by Tian and Pearl [2].

All current algorithms for performing causal inference follow the *single agent* or centralized paradigm. This means that for each application domain there is one single agent using a single model for the entire domain. There are several reasons why this is a weakness for the applicability of the current state-of-the-art algorithms, such as:

- Large and complex models are difficult to learn, handle and maintain.
- Some domains are inherently distributed (e.g., the internet), trying to solve them with a centralized approach could introduce a communications bottleneck.
- Knowledge can be distributed over different sources with partial knowledge and some of them might not necessarily want to disclose all of their information.

Therefore, we propose a *cooperative multi-agent* approach to causal inference, where each agent models overlapping parts of the domain and then cooperates with the other agents to perform causal inference. The multi-agent paradigm is promising for overcoming the limitations of the single agent paradigm for the following reasons:

- They allow to model complex knowledge in small modular components.
- It is a natural way to model inherently distributed problem domains.
- In some cases it is possible to perform tasks in cooperation with other agents while only disclosing a part of their information, and thus allowing sensitive information to remain private.

[1] With unobserved variables we mean variables whose values are never observed to the modeler.

In this article, we extend causal Bayesian networks to *multi-agent causal models*. Such a model consists of a collection of agents that each contain a graphical model over their domain variables. These variables are of two types: *private* variables that remain undisclosed to other agents, and *shared* variables whose values and distributions are accessible for other agents. Furthermore, we will introduce an algorithm for performing causal inference in multi-agent causal models. In the algorithm only information concerning shared variables is exchanged between agents.

Causal inference can be impossible in a multi-agent causal model for two reasons. One is because in some cases causal inference is impossible due to the fact that there are too many hidden variables in the single agent model that would be obtained by seeing all variables as shared variables.[2] The other reason is that in some cases the set of shared variables of some agents is too small to be able to perform causal inference. While for the former problem there is no solution, for the latter there is. In previous work [3] we have developed a framework where the agents can negotiate with each other to disclose some extra private variables, if this does not interfere with their privacy concerns, so that multi-agent causal inference might be possible.

To summarize, our approach consists of four phases:

(1) The validation of some assumptions needed for our approach to work.
(2) Checking identifiability, this is investigating whether causal inference is possible in principle (i.e. whether there are not too many unobserved variables).
(3) Negotiation and disclosure, this phase is only initiated when causal inference is not possible because too many variables are private in individual agents. The agents negotiate and if they reach an agreement on which variables to disclose, then they disclose them and update their joint probability distribution.
(4) Actual computation of causal queries, also called identification. In this final phase, the effect of manipulating some variables on other variables is computed.

In this article, we will only briefly touch upon the first three phases and mostly focus on the actual computation of causal effects. Furthermore, for the sake of clarity we will limit ourselves to the specification of algorithms for models with two agents. For investigations of models organised in a chain we refer to previous work [4,3]. Finally, we stress that the agents are cooperative in the sense that they are truthful to each other and work together to solve a problem.

In the following two sections we will review causal Bayesian networks and causal inference from a single agent perspective. Next, we will introduce multi-agent causal models, followed by a section with the different steps of our multi-agent causal inference algorithm. Finally, we will conclude with a discussion and indicate some points for possible future research.

2. Causal graphical models

In this section, we introduce causal Bayesian networks and contrast them with classical probabilistic Bayesian networks.

[2] Remark that we can infer this information from the multi-agent model without trespassing on the privacy of the individual models.

In this work uppercase letters are used to represent variables or sets of variables, while corresponding lowercase letters are used to represent their instantiations. $P(X = x)$ is used to denote the probability distribution over the values of variable X to value x. Usually, $P(x)$ is used as an abbreviation of $P(X = x)$. $Ch(X)$, $Pa(X)$ and $Anc(X)$ respectively denote the children, parents and ancestors of variable X in the graph. Furthermore, $Pa(x)$ represents the values of the parents of X.

2.1. Bayesian networks

The classical *Bayesian network* (BN) is a concise representation of a joint probability distribution (JPD) $P(v)$ over a set of random variables $V = \{V_1, \ldots, V_n\}$. In [5,6] they are defined as follows:

Definition 2.1. A Bayesian network is a triple $\langle V, G, P(v_i|Pa(v_i))\rangle$, with:

- $V = \{V_1, \ldots, V_n\}$, a set of observable discrete random variables.
- A directed acyclic graph (DAG) G, where each node represents a variable from V.
- Conditional probability distributions (CPD) $P(v_i|Pa(v_i))$ of each variable V_i from V conditional on its parents in the graph G.

See Fig. 1 for a famous example adopted from [5], representing an alarm system. The alarm can be triggered either by a burglary, by an earthquake, or by both. The alarm going of might cause John and/or Mary to call the house owner at his office.

The directed edges in the graph G can be seen as probabilistic dependencies between the corresponding variables, while the missing edges represent conditional independencies between variables via the *Markov property* [7]: Every variable is independent of any of its non-descendants given its immediate parents in the graph. It is the incorporation of such independence relations that makes a BN a more concise representation than the full JPD.

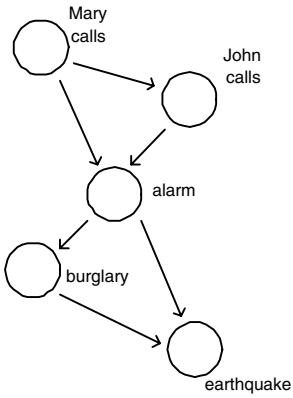

Fig. 1. Example of a Bayesian network representing an alarm system.

The Markov property implies that the CPDs of a BN represent a factorization of the joint probability distribution as a product of conditional probability tables of each variable given its parents in the graph:

$$P(v) = \prod_{V_i \in V} P(v_i | Pa(v_i)) \tag{1}$$

Other, more complex conditional independence properties are also implied by the structure of the graph, and can be derived using the *d-separation* criterion [1].

Bayesian networks allow to efficiently answer probabilistic queries such as $P(burglary = true | Johncalls = true, Marycalls = false)$, i.e. the probability that there was a burglary, given that we know John called and Mary did not. See [8] for an overview.

Remark that a single JPD can be represented by several BNs, by using different subsets of all the conditional independence assertions present in the JPD.

2.2. Causal Bayesian networks

A *causal Bayesian network* (CBN) is a Bayesian network $\langle V, G, P(v_i|Pa(v_i))\rangle$ with some extra properties. The basic definition remains the same as for classical BNs and the Markov property and the factorization from Eq. (1) still hold.

What differs CBNs from classical BNs is that the directed edges are viewed as representing *autonomous causal relations* among the corresponding variables, while in a BN the directed edges only represent a probabilistic dependency, and not necessarily a causal one.

For a relation from variable C to variable E to be causal, must be understood as the presence of an effect that would occur in variable E after a manipulation of variable C. More specifically, when one would vary C by performing a surgical manipulation on variable C, i.e. a manipulation that only changes C all else staying equal, then variable E would also vary.

This means that in a CBN, each CPD $P(v_i|Pa(v_i))$ represents a stochastic assignment process by which the values of V_i are chosen in response to the values of $Pa(V_i)$. This is an approximation of how events are physically related with their effects in the domain that is being modeled. Furthermore, these assignment processes are assumed to stay invariant under variations in the processes governing other variables [1].

In the BN of Fig. 1, these assumptions clearly do not hold, for example in the underlying physical domain, whether or not there is an earthquake is not caused by the state of the variables alarm and burglary.

In Fig. 2, we see a causal BN that represents the same JPD as the BN in Fig. 1. Here the extra assumptions do hold. E.g., we can fathom that the state of the alarm can be caused by a burglary, an earthquake or both, and thus the CPD $P(alarm|burglary, earthquake)$ represents an assignment process that is an approximation of how the alarm is physically related to earthquake and burglary. Moreover, if the sensitivity of the alarm system is changed, this will only imply a change in the process $P(alarm|burglary, earthquake)$, but not in the processes governing other variables.

Just as in classical BNs, probabilistic inference is possible in CBNs, furthermore, the extra assumptions allow to do causal inference, i.e. to predict the effect of external manipulations or interventions of some variables. See the next section for more on this.

The structure of BNs and causal BNs can be obtained via experts, learning methods based on observational or experimental data, or a combination of these.

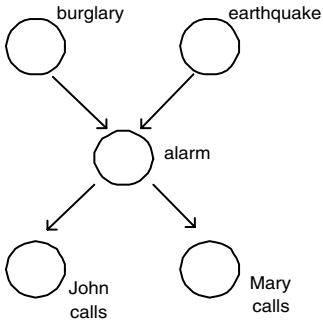

Fig. 2. A causal BN that represents the same JPD as the BN in Fig. 1.

2.3. Semi-Markovian causal models

In this section, we drop the simplifying assumption that there are no unobserved variables and introduce models where unobserved variables can be explicitly included, but where these have special properties.

Definition 2.2. A semi-Markovian causal model (SMCM) [9] is a causal Bayesian network where some of the variables are unobserved, and furthermore, every unobserved variable has no parents (i.e. is a root node) and has exactly two observed children.

An example of a semi-Markovian model with two unobserved variables is shown in Fig. 3. We sometimes represent unobserved variables by a dashed bi-directed edge.

In SMCMs there are observed variables $V_1, \ldots, V_n$ and unobserved variables $U_1, \ldots, U_k$. Here, the observed probability distribution $P(v)$ can be written as a product over the observed variables given their observed and unobserved parents and a product over the marginal of the unobserved variables, as in a SMCM the unobserved variables have no parents by definition. There is also a summation over all the values of the unobserved variables U to obtain the JPD over the observed variables. The equation becomes

$$P(v) = \sum_{\{u_l | U_l \in U\}} \prod_{V_i \in V} P(v_i | Pa(v_i), UPa(v_i)) \prod_{U_j \in U} P(u_j) \tag{2}$$

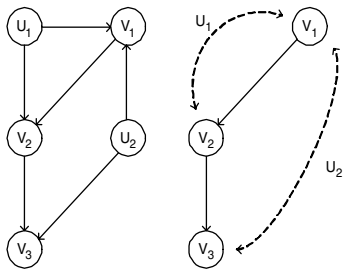

Fig. 3. An example semi-Markovian causal model, on the left the full notation and on the right the notation with bi-directed edges.

where we sum over all values u_l of all unobserved variables U_l and where $Pa(\,)$ and $UPa(\,)$ stand for the sets of the observed and unobserved parents, respectively.

The importance of SMCMs stems from the fact that CBNs with arbitrary sets of unobserved variables can be converted into a semi-Markovian model while preserving the causal inference properties [10]. This allows to develop causal inference algorithms for the simpler class of SMCMs that can be used for every CBN. When confronted with a CBN with arbitrary unobserved variables, it is first converted into a SMCM before applying a causal inference algorithm, see [2] for the transformation algorithm.

3. Single agent causal inference

In this section, we introduce the problem of causal inference. We start by explaining the difference between observing and intervening or manipulating, then treat an important theorem that specifies how to incorporate the effect of a manipulation in a CBN. After that the problem of identification is explained, followed by the introduction of a state-of-the-art causal inference algorithm due to Tian and Pearl.

3.1. Observation vs. manipulation

An important issue in graphical models is to distinguish between different types of conditioning, each of which modify a given probability distribution in response to information obtained.

Conditional probabilities are defined as

$$P(Y = y|X = x) = P(y|x) = \frac{P(Y = y, X = x)}{P(X = x)} \tag{3}$$

This is what is referred to as *conditioning by observation*. It represents the way in which a probability distribution of Y should be modified when a modeler passively observes the information $X = x$.

It is important to realize that this is typically not the way the distribution of Y should be modified if we intervene externally and force the value of X to be equal to x. We refer to this type of modification as *conditioning by intervention* or *manipulation*.[3] To make the distinction clear, Pearl has introduced the *do-operator* [1][4]:

$$P(Y = y|X = do(x)) \tag{4}$$

Generally the two quantities will be different

$$P(Y = y|X = do(x)) \neq P(Y = y|X = x) \tag{5}$$

and the quantity on the left-hand side cannot be calculated from the joint probability distribution $P(v)$ alone, without additional assumptions imposed on the graph, i.e. that the directed edges have a causal meaning and are autonomous.

Consider the simple CBN consisting of two variables: $X \rightarrow Y$. In this case $P(y|do(x)) = P(y|x)$ as X is the only immediate cause of Y, but $P(x|do(y)) = P(x) \neq P(x|y)$ as there is no causal path from Y to X.

[3] Throughout this text the terms *intervention* and *manipulation* are used interchangeably.

[4] In the literature other notations such as $P(Y = y \| X = x)$, $P_{X=x}(Y = y)$, or $P(Y = y|X = \hat{x})$ are abundant.

3.2. Manipulation theorem

Performing an external manipulation in a domain that is modeled by a CBN, changes that domain and the JPD that is used to model it. Here we introduce a theorem that specifies how a CBN and the JPD that is associated with it, must be changed to incorporate the change induced by an external manipulation.

For reasons of simplicity we only study the simplest form of interventions in this work, i.e. fixing a set of variables T to some constants $T = t$. In principle more general types of intervention are possible, such as fixing the value of variables T in a way that depends on previously observed variables. Furthermore, we only take into account interventions on observable variables.

Due to lack of space, we immediately introduce the theorem for semi-Markovian causal models. In that case the factorization of the JPD consists of a part with observable variables V and one with unobservable variables U.

To obtain the post-intervention distribution after fixing a set of variables $T \subset V$ to a fixed value $T = t$, the factors with the variables in T conditional on their parents in the graph (i.e. their causes in the pre-intervention distribution), have to be removed from the JPD. As we only treat interventions on observable variables, the factors that have to be removed are in the part over the observable variables V. Formally, these factors are: $P(t_i|Pa(t_i), UPa(t_i))$ for all variables $T_i \in T$.

This is because after the external intervention, it is the intervention $T_i = t_i$ rather than the parent variables in the graph that cause the values of the variables in T. Furthermore the remaining occurrences of T in the JPD have to be instantiated to $T = t$.

See Fig. 4 for an example manipulation in a SMCM. After a manipulation of variable V_2 to a value v_2 the parent relations of this variable are removed from the graph. Note that the confounding variable U_1 disappears from the graph altogether, as after the manipulation it is no longer a confounding factor.

A manipulation of this type only has a local influence in the sense that only the incoming links of a manipulated variable have to be removed from the model, no factors representing other links have to be modified, except for instantiating the occurrences of the manipulated variables T to t. This is a consequence of the assumption of CBNs that the factors of the JPD represent assignment processes that must stay invariant under variations in the processes governing other variables (see Section 2.2). This leads to the formal theorem:

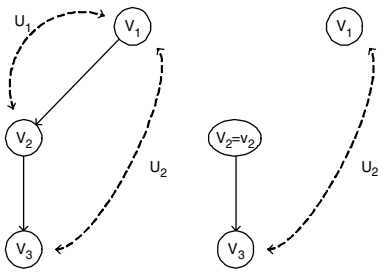

Fig. 4. On the left a semi-Markovian causal model, on the right the graph obtained after manipulating variable V_2 to a value v_2.

Theorem 3.1. *Given a SMCM with observable variables* $V = V_1, \ldots, V_n$ *and unobservable variables* $U = U_1, \ldots, U_{n'}$ *and we perform the manipulation* $T = t$ *for a subset of variables* $T \subseteq V$*, the post-manipulation distribution becomes*:

$$P(v|do(t)) = \sum_{\{u_I|U_I \in U\}} \prod_{V_i \in V \setminus T} P(v_i|Pa(v_i), UPa(v_i)) \prod_{U_j \in U} P(u_j) \Bigg|_{T=t} \tag{6}$$

Generally we are only interested in the post-intervention distribution over a subset $S \subset V$ of the observed variables. To obtain this we simply have to marginalize over the unwanted variables after applying the manipulation theorem:

$$P(s|do(t)) = \sum_{V_j \in V \setminus (S \cup T)} P(v|do(t)) \tag{7}$$

3.3. Identification of causal effects

Before explaining the algorithm, we introduce some basic concepts.

Definition 3.2. The *causal effect* of a variable X on a set of variables S is the response of variables S to intervention on variable X and, as mentioned previously, is noted as $P(s|do(x))$.

Definition 3.3. A causal effect is *identifiable* from a SMCM if the quantity $P(s|do(x))$ can be computed, uniquely using values from the observed distribution $P(v)$, and thus not needing the distributions of unknown variables.

E.g., in a SMCM the causal effect $P(v|do(x))$ of one variable X on all the others is identifiable if in Eq. (6) of the manipulation theorem, the factors including unobserved variables such as $P(u_j)$ and $P(v_i|Pa(v_i), UPa(v_i))$ where $UPa(v_i)$ is non-empty, can be omitted and thus $P(s|do(x))$ can be calculated from the JPD $P(v)$ over the observed variables.

The process of calculating a causal effect from a causal graphical model is what we call *causal inference* or *identification*.

As previously mentioned, we will work in the family of semi-Markovian causal models (SMCM) because they are simpler than CBNs with arbitrary unobserved variables. Furthermore, such a complicated CBN can be converted into a SMCM while preserving the same properties for performing causal inference.

In general when in a SMCM no unobserved variables are connected to the manipulated variable X, $P(v|do(x))$, or the causal effect of X on every other variable, is identifiable from the JPD of the observed variables $P(v)$. This result follows immediately from the manipulation theorem.

3.4. Single agent identification algorithm

In this section, we discuss a structured way to reason about identification and introduce an algorithm for calculating causal effects such as $P(y|do(x))$ from an arbitrary SMCM if this is possible.

As we have seen before, a causal effect is identifiable from a CBN if it can be calculated from the distribution of the observed variables V alone.

Given a SMCM over observable variables V and unobservable variables U, at the start the identification formula obtained via the manipulation theorem refers to all elements in both V and U. The goal of an identification algorithm is to gradually remove the influence of the unobserved variables U from the formula, to finally obtain a distribution over observed variables from which the desired causal effect can easily be calculated.

The algorithm that we will discuss here, was first introduced by Tian and Pearl in [2] and it represents the state-of-the art in causal inference at this moment. It is based on a new factorization of the JPD of an SMCM into so-called c-factors. Before going into the details of the algorithm we introduce these new concepts.

3.4.1. C-components and c-factors

Let a path entirely composed of bi-directed edges be called a *bi-directed path*.

Definition 3.4. In a semi-Markovian causal model, the set of observable variables can be partitioned into disjoint groups by assigning two variables to the same group iff they are connected by a bi-directed path. We call such a group a *c-component* (from "confounded component") [2].

Variables that are not connected to any bi-directed edge are a c-component by themselves. All c-components of a SMCM are disjoint and constitute a partition.

Furthermore, when we say that unobservable variables belong to the same group if and only if they are part of bi-directed paths connecting two variables belonging to the same c-component, this also induces a partition on the unobserved variables.

Consider for example the SMCM of Fig. 5, it is partitioned into five c-components $S_1, \ldots, S_5$:

$$\{\{X, X_4, X_6\}, \{X_1\}, \{X_2, X_5\}, \{X_3\}, \{Y\}\} \tag{8}$$

The associated partition $N_1, \ldots, N_5$ of the unobserved variables is as follows:

$$\{\{U_1, U_2\}, \emptyset, \{U_3\}, \emptyset, \emptyset\} \tag{9}$$

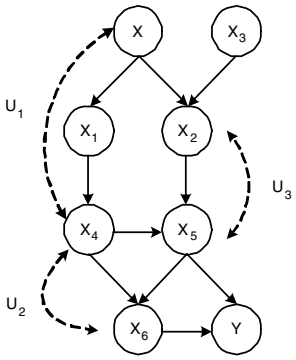

Fig. 5. A SMCM with five c-components.

Now assume that V is partitioned into its k c-components $S_1, \ldots, S_k$ and denote by N_l the set of unobservable variables that are parents of variables in S_l. As stated above, $N_1, \ldots, N_k$ also form a partition of U.

Definition 3.5. A *c-factor* $Q[S]$ of a set of observable variables $S \subset V$ is the contribution of the variables in S and the associated unobserved variables N (those that are connected to variables in S), to the mixture of products representing the JPD of a SMCM (Eq. (2)):

$$Q[S] = \sum_{\{u_l | U_l \in N\}} \prod_{V_i \in S} P(v_i | Pa(v_i), UPa(v_i)) \prod_{U_j \in N} P(u_j) \tag{10}$$

If we apply this definition to the c-components $S_1, \ldots, S_k$ of an SMCM, then the disjointness of these c-components and their associated unobservables $N_1, \ldots, N_k$ implies that the mixture of products for $P(v)$ of Eq. (2) can be decomposed into a product of c-factors $Q[S_l]$ of the c-components:

$$P(v) = \sum_{\{u_m | U_m \in U\}} \prod_{V_i \in V} P(v_i | Pa(v_i), UPa(v_i)) \prod_{U_j \in U} P(u_j) \tag{11}$$

$$= \prod_{l=1}^{k} Q[S_l] \tag{12}$$

This result implies that the mixture of products of Eq. (2) can be transformed into a factorization. That is useful as a factorization allows a calculation to be split up into a product of different modular subcalculations, while when confronted with a mixture of products this is not necessary possible.

Furthermore, when applying the manipulation theorem for SMCMs of Eq. (6), we see that the c-factor of a c-component S_l is also the post-intervention distribution of the variables in S_l, under an intervention that sets all other observable variables $V \setminus S_l$ to constants, or

$$Q[S_l] = P(s_l | do(v \setminus s_l)) \tag{13}$$

Finally, it follows from the definition that

$$P(v) = Q[V] \tag{14}$$

As an illustration, in the SMCM of Fig. 5 some of the c-factors are[5]:

$$Q[X, X_4, X_6] = \sum_{u_k \in \{U_1, U_2\}} P(x|u_1) P(x_4|x_1, u_1, u_2) P(x_6|x_4, x_5, u_2) \cdot P(u_1) P(u_2) \tag{15}$$

$$Q[X_1] = P(x_1|x)$$

3.4.2. Important lemmas

The identification algorithm that we discuss is based on three important lemmas that will be briefly introduced in this section. See [2] for a complete specification of the lemmas and their proofs.

[5] For notational convenience we will write $Q[X, Y, Z]$ instead of $Q[\{X, Y, Z\}]$.

First some notation: given a SMCM with graph G and observable variables V and unobservable variables U, then if $W \subset V$, G_W is the subgraph of G restricted to variables W and the associated unobservable variables. See Fig. 6 for an example reduction of the graph in Fig. 5. We also introduce the notation $Anc(X)_G$, this denotes the ancestors of X in the graph G. Furthermore, a topological order over V is noted as $V_1 < \cdots < V_n$, and let $V^{(i)} = \{V_1, \ldots, V_i\}$, $i = 1, \ldots, n$, and $V^{(0)} = \emptyset$.

Lemma 3.6. *The c-factor $Q[S]$ of every c-component S in a SMCM with observable variables V is identifiable from the JPD $P(v)$ as follows*:

$$Q[S] = \prod_{V_i \in S} P(v_i | Pa(T_i) \setminus \{v_i\}) \tag{16}$$

where T_i is the c-component of $G_{V^{(i)}}$ that contains V_i.

This lemma implies that every c-factor $Q[S]$, and thus the post-intervention distribution of S after manipulating $V \setminus S$, can always be calculated from the JPD $P(v)$. From Eq. (16) we can see that the variables that are needed for calculating the c-factor $Q[S]$ of a c-component S are $S \cup Pa(S)$.

Lemma 3.7. *In a SMCM with graph G over observable variables V, where $A \subset B \subset V$, if A is an ancestral set in the graph G_B ($A = Anc(A)_{G_B}$), then $Q[A]$ is identifiable from $Q[B]$ as follows*:

$$Q[A] = \sum_{V_i \in B \setminus A} Q[B] \tag{17}$$

The lemma implies that when we have the c-factor $Q[B]$ over some set B (e.g., via Lemma 3.6), and a subset of B is ancestral in the graph reduced to variables B, then $Q[A]$ can be calculated from $Q[B]$, and this simply by marginalizing over $Q[B]$.

For example, imagine the SMCM of Fig. 5 reduced to S_X, the c-component of X. The set (X_4, X_6) is ancestral in this graph, as $Anc(X_4, X_6)_{G_{S_X}} = (X_4, X_6)$, and thus Lemma 3.7 implies that $Q[X_4, X_6]$ can be calculated from $Q[X, X_4, X_6]$ via:

$$Q[X_4, X_6] = \sum_X Q[X, X_4, X_6]$$

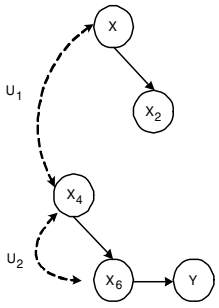

Fig. 6. The SMCM of Fig. 5 reduced to the observed variables $W = \{X, X_2, X_4, X_6, Y\}$ and the associated unobserved variables $\{U_1, U_2\}$.

When we apply this to the c-factor of $Q[X, X_4, X_6]$, as calculated previously (Eq. (15)), we get

$$\begin{aligned}\sum_X Q[X, X_4, X_6] &= \sum_X \sum_{U_1,U_2} P(x|u_1)P(x_4|x_1,u_1,u_2)P(x_6|x_4,x_5,u_2)\cdot P(u_1)P(u_2)\\ &= \sum_{U_1,U_2} P(x_4|x_1,u_1,u_2)P(x_6|x_4,x_5,u_2)P(u_1)P(u_2)\\ &= Q[X_4, X_6]\end{aligned}$$

We see that the factor $P(x|u_1)$ can be marginalized away, because that is the only occurrence of X, and then we obtain exactly $Q[X_4, X_6]$.

Lemma 3.8. *In a SMCM with graph G over observable variables V, where $H \subset V$ and H is partitioned in c-components $H_1, \ldots, H_l$ in G_H, then*

$$Q[H] = \prod_{i=1}^{l} Q[H_i] \tag{18}$$

Furthermore, every $Q[H_i]$ is identifiable from $Q[H]$.

Finally, this lemma resembles Lemma 3.6, but instead of treating c-components of entire graphs it can be used to calculate the c-components of a subgraph G_H.

3.4.3. Sketch of algorithm

In this section, we give an intuitive sketch of the identification algorithm. For simplicity we will treat the problem of calculating $P(y|do(x))$, where both X and Y are single variables. The algorithm can very easily be extended for the calculation of causal queries such as $P(s|do(x))$, where S is a set of variables, extending it to the case where more than one variable is being manipulated poses somewhat more problems. See [2] for all the details.

Following the manipulation theorem from Section 3.2, to obtain the effect of $P(y|do(x))$, the factor $P(x|Pa(x), UPa(x))$ has to be removed from the JPD. Furthermore, in Section 3.4.1 we introduced a factorization of the JPD of a SMCM into c-factors of c-components. The factor that has to be removed is part of the c-factor of the c-component containing variable X. We will denote this c-component by S_X and the c-factor by $Q[S_X]$. This gives

$$P(y|do(x)) = \sum_{V\setminus Y} Q[S_X \setminus \{X\}] \prod_i Q[S_i] \tag{19}$$

From Lemma 3.6 in the previous section, we know that all the c-factors of the c-components $S_X, S_1, \ldots, S_k$ can be computed from the JPD $P(v)$. But, as X might have unobserved parents $UPa(X)$, we do not necessarily have explicit access to $P(x|Pa(x), UPa(x))$.

Therefore, we will transform our problem slightly. Define $D = Anc(Y)_{G_{V\setminus\{X\}}}$, or the ancestors of Y in the graph without X, and thus by Lemma 3.7:

$$P(y|do(x)) = \sum_{D\setminus Y} Q[D] \tag{20}$$

Also define $D_X = D \cap S_X$, and, $D_i = D \cap S_i$, for all $i = \{1, \ldots, k\}$. As D is an ancestral set in $G_{V \setminus X}$, each D_i will also be ancestral in its respective graph G_{S_i} and can thus be calculated from $Q[S_i]$ by applying Lemma 3.7.

On the other hand D_X, will not necessarily be ancestral in G_{S_X} as X can be an ancestor of D_X. Now we can write $Q[D]$ as follows:

$$Q[D] = Q[D_X] \prod_i Q[D_i] \tag{21}$$

At this point, we have transformed our problem in calculating $Q[D_X]$ from $Q[S_X]$.

The original graph reduced to these variables D_X and S_X is sufficient to know whether a causal effect $P(y|do(x))$ can be computed from the JPD over the observed variables V. To actually compute $P(y|do(x))$, all observable variables V of the JPD are needed, but for the variables $V \setminus D_X$, the calculations are trivial.

Now, we will try to calculate $Q[D_X]$ from $Q[S_X]$. In the case that D_X is not an ancestral set in G_{S_X}, its c-factor $Q[D_X]$ cannot be calculated as a whole from $Q[S_X]$. Instead, we rewrite $Q[D_X]$ as a factorization of the c-factors of its c-components $D_{X,j}$ in the graph G_{D_X}:

$$Q[D_X] = \prod_j Q[D_{X,j}]$$

Now if we can calculate each $Q[D_{X,j}]$, our problem is solved. To do this we proceed as follows: each of these $D_{X,j}$ is either

(1) ancestral in G_{S_X}, or $Anc(D_{X,j})_{G_{S_X}} = D_{X,j}$
(2) its ancestral set is equal to S_X itself, or $Anc(D_{X,j})_{G_{S_X}} = S_X$
(3) its ancestral set is strictly in between itself and S_X, or $D_{X,j} \subset Anc(D_{X,j})_{G_{S_X}} \subset S_X$.

In the first case we can use Lemma 3.7 to obtain $Q[D_{X,j}]$ by marginalizing over $Q[S_X]$. In the second case, $Q[D_{X,j}]$ is not identifiable in this SMCM and consequently, $P(y|do(x))$ is not identifiable. In the final case, $D_{X,j}$ belongs to a single c-component H_X in the graph reduced to its ancestral set A, and using Lemmas 3.7 and 3.8, $Q[H_X]$ can be obtained from $Q[S_X]$. Then the process restarts, but now we want to calculate $Q[D_{X,j}]$ from $Q[H_X]$ instead of $Q[S_X]$.

These steps have to be repeated until we reach one of the first two cases for each $Q[D_{X,j}]$. When all $Q[D_{X,j}]$ are identifiable (i.e. each process ends in case 1), the desired quantity $P(y|do(x))$ is identifiable using the equations above. As soon as one of the $Q[D_{X,j}]$ is not identifiable (i.e. the process ends in step 2), $P(y|do(x))$ is not identifiable in the given SMCM.

In Fig. 7, we see a conceptual sketch of the identification algorithm. The first step is to calculate $Q[S_X]$ from the JPD $P(V)$ using Lemma 3.6. Then, we try to calculate each of the $D_{X,j}$ from $Q[S_X]$. This is done by finding its ancestral set A in the subgraph G_{S_X}. Depending on the composition of this set A, we distinguish between three cases:

(1) The set $D_{X,j}$ has no external ancestors in G_{S_X}, then Lemma 3.7 can be used to calculate $Q[D_{X,j}]$.
(2) The ancestral set A of $D_{X,j}$ in G_{S_X} is equal to the complete set S_X, in this case the desired causal effect is not identifiable.

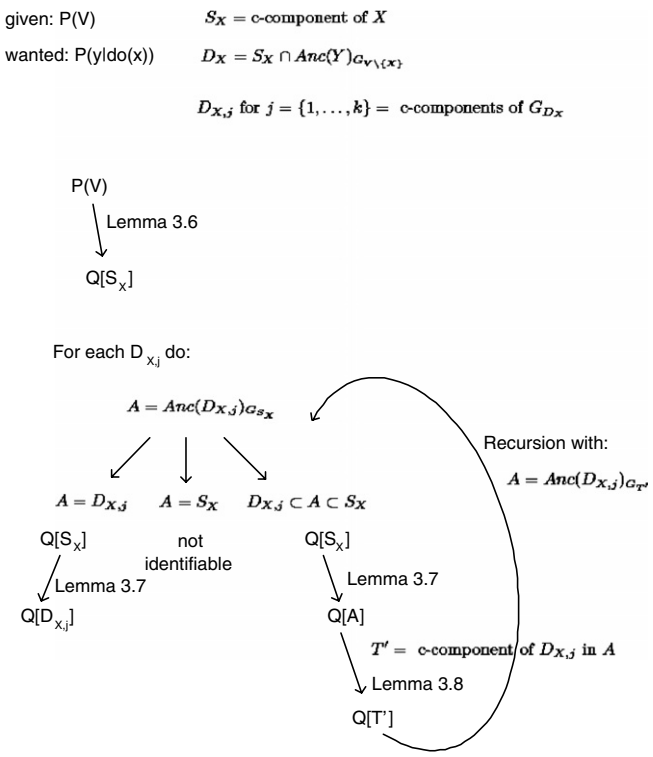

Fig. 7. A conceptual sketch of the identification algorithm.

(3) The ancestral set A lies in between $D_{X,j}$ and S_X, then using Lemmas 3.7 and 3.8 can be used to reduce the problem to calculating $Q[D_{X,j}]$ from a strict subset of S_X.

This process is repeated until one of the first two cases is reached.

4. Cooperative multi-agent systems

In the previous sections of this article, there was the silent assumption that there is a single computational entity or agent that can model the entire domain. In reality however, this assumption rarely materializes. As domains become larger, more complex, open, and distributed, building and maintaining a model by a single agent, limited by its knowledge, perspective, and computational resources, for modeling such domains, would prove to be very costly or even impossible. Instead a *cooperative multi-agent system* is needed to address the modeling task effectively, consisting of a set of cooperating agents, where each agent has partial knowledge about the domain, and can observe the domain from a partial perspective. Just as in single agent systems, an agent can reason and act autonomously, but to overcome its limit in domain knowledge, an agent additionally can benefit from other agents' knowledge, perspectives, and computational resources through communication and coordination.

Applying the multi-agent paradigm to graphical models means that there is no more need for the assumption that there is one single agent observing or not observing (and thus modeling or not modeling) all the variables in the domain. Instead there are several agents who each observe and model a subset of all the variables. Agents that share variables in their models can then communicate (some of) their knowledge on these shared variables to each other.

See [11] for an introduction to multi-agent systems and [12] for a collection of recent advances in the area.

4.1. Advantages

In what follows we enumerate and discuss the advantages of the multi-agent paradigm:

- In large and complex domains, diverse knowledge is required. Instead of combining all this into a vast model, it is better to handle this complexity and diversity by organising the knowledge in a *modular* manner, so that every modular entity can be modeled by a relatively lightweight single agent. This approach simplifies both development and upkeep.
- Components of a complex system are often *distributed*, as are sensors providing observations of such systems. Traditionally, the observations from distributed sensors are transmitted to a central site, where a single agent processes them, decides on necessary actions and transmits control signals to the locations where the actions take place. Transmission of observation and action control signals, however is hampered by limited communication bandwidth, communication delay and potential communication failure. The multi-agent approach suggest to deploy multiple agents near a component (or small group of nearby components), allowing the sensor outputs to be processed on site and actions to be taken more promptly.
- Many complex domains are often *open-ended*. As in a multi-agent system, the knowledge and decision making relative to single or small groups of components, is captured into a separate agent, the addition of new components can be handled by dynamically adding agents and letting existing agents adapt to the new agent community.

4.2. State-of-the-art

Now we introduce some state-of-the-art methods in multi-agent graphical modeling that can be viewed as related work to our approach.

Multiply sectioned Bayesian networks (MSBN) are a knowledge representation formalism for cooperative multi-agent reasoning under uncertainty developed by Xiang [13]. There exist extensions of the junction tree inference algorithm to perform multi-agent probabilistic inference. In a MSBN the edges are not necessarily causal (i.e. the manipulation theorem does not hold) and the agents' local models are purely Markovian (i.e. they do not take into account hidden variables or confounders). The consequence of this is that in MSBNs causal inference is not possible.

Distributed perception networks (DPN) are an agent-based approach to fusion of heterogeneous data and information introduced by Pavlin et al. [14,15]. DPNs focus on

calculating the marginal probability for a single hypothesis node based on large amounts of evidence. For the same reasons as in MSBNs, performing causal inference is not possible in DPNs.

Qualitative multi-agent causal reasoning is an approach introduced by Chaib-Draa based on *causal maps*, a type of cognitive map that focuses on influence, causality and system dynamics [16]. They focus on qualitative reasoning, and quantitative causal inference is not possible in these models.

Object oriented Bayesian networks (OOBN) approach the modeling of complex domains with BNs by describing such a domain as inter-related objects, with object-oriented properties [17]. Again, the edges between variables are not necessarily causal and semi-Markovian models are not allowed, thus making kind of causal inference we are interested in impossible.

5. Multi-agent causal models

In this section, we introduce multi-agent causal models (MACM) [18]. It is assumed that there is no longer one central controller having access to all the observable variables, but instead a collection of agents each having access to non-disjoint subsets V_i of all the variables V.

5.1. Definition

Definition 5.1. A *multi-agent causal model* consists of n agents, each of which contains a semi-Markovian model M_i:

$$M_i = \langle V_{M_i}, G_{M_i}, P(V_{M_i}), K_{M_i} \rangle \quad \text{for } i \in \{1, \ldots, n\}$$

- V_{M_i} is the subset of variables agent-i can access.
- G_{M_i} is the causal graph over variables V_{M_i}.
- $P(V_{M_i})$ is the joint probability distribution over V_{M_i}.
- K_{M_i} stores the intersections V_{M_i,M_j} with other agents j, $\{V_{M_i} \cap V_{M_j}\}$. We assume that the agents agree on the structure and the distribution of their intersections.

Furthermore, for an MACM to be valid it is assumed that:

- No edges (either directed or bi-directed) are allowed between variables of different agents i and j, except if both belong to the intersection V_{M_i,M_j}.
- The intersection between two agents are modeled in the same way.
- Combining the models of the individual agents does not introduce directed cycles.

The assumption of acyclicity is not always trivial to check in a distributed fashion, for that task will use an adaptation of Xiang's algorithm for the distributed verification of acyclicity [13].

The set of variables of an agent-i without all the intersections he shares with other agents is called the set of *private variables*. Also, the union of the graphs G_{M_i} of all agents in the MACM is noted as $\cup G$, i.e. the graph that would be associated with a single agent

having access to the entire domain. Likewise, the union of all variables V_{M_i} is denoted by $\cup V$.

5.2. Example

An example MACM adapted from [19] is depicted in Fig. 8. In this case $V_{M_1} = \{X, X_1, \ldots, X_9\}$ and $V_{M_2} = \{X_7, \ldots, X_{11}, Y\}$, while $V_{1,2} = \{X_7, X_8, X_9\}$. With this model a company wants to assess the influence of its *product pricing* strategy on *product decision*, this is whether a new product is launched or not. Furthermore this company consists of two divisions, one modeled by agent1, roughly responsible for external issues such as the market situation or the competitive strategy, and one modeled by agent2 pertaining to internal issues such as research and development and the rate at which new products are launched.

5.3. Structure learning

Just as a classical Bayesian network, a MACM can be learnt from data, an expert or both.

In other work, we have developed an approach for learning a MACM from distributed heterogeneous data with overlap between the data at the different sites [20,3].

It is a constraint-based approach based on finding conditional independencies in the data. Each agent has only access to data concerning variables in the site he is responsible for.

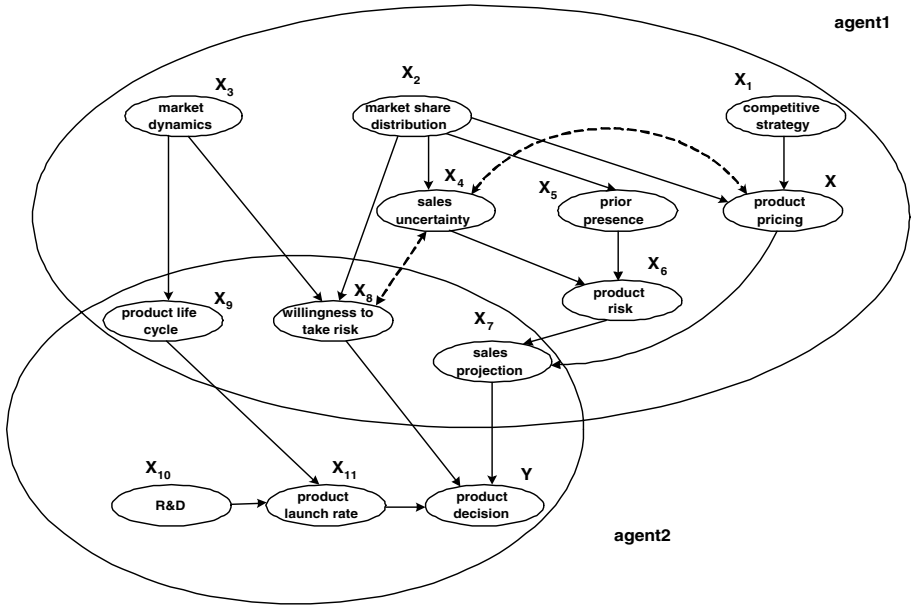

Fig. 8. Example of a multi-agent causal model of a product decision model.

Special attention is devoted to finding so-called *g-false dependencies*. These are variables in an agent that appear to be dependent from the point of view of that agent, but that are actually independent given some variables in another agent. We offer solutions for solving this problem while exchanging a minimum amount of information between the different sites.

In the rest of this work we assume that the MACM representing the data is given.

6. Overview of multi-agent identification algorithm

In this section, a schematic introduction of our algorithm for performing causal inference in MACMs, introduced in [21–23], is given. For reasons of convenience we will focus on the identification of $P(y|do(x))$ where both X and Y are atomic variables.

Furthermore, we will mainly focus on algorithms for MACMs consisting of two agents. Such models are called *bi-agent causal models* (M_1, M_2), where $M_i = \langle V_{M_i}, G_{M_i}, P(V_{M_i}), K_i \rangle$ for $i \in \{1,2\}$, where agent1 typically contains the intervention variable X and agent2 contains the variable to be studied Y. In [4,3] we extended our algorithms to chain MACMs, consisting of more than two agents organised in a chain.

For each causal query $P(y|do(x))$ the following steps are taken, each of which will be explained next.

(1) Identifiability.
(2) Negotiation.
(3) Identification of $P(y|do(x))$.

6.1. Identifiability

In the first of the three phases it is investigated whether a specific query of the form $P(y|do(x))$ would be identifiable from the underlying single agent model.

In [18] we introduced an algorithm for performing this task for $P(y|do(x))$ in multi-agent causal models where Y is in the intersection, based on methods developed by Tian in [2]. The algorithm only uses communication between agents concerning structural information of variables in their intersection and the variables X and Y, thereby ensuring that the privacy of each agents' private model is protected. Or in other words, to know whether in a MACM a causal query $P(y|do(x))$ is identifiable in the underlying single agent model, we do not need to construct this single agent model, but can infer the identifiability from the MACM solely by communicating structural information concerning the intersections between the agents.

In [3] we have extended this algorithm to models where Y does not have to be in the intersection. Due to space constraints and because all the steps of the algorithm are implicit in the actual identification phase (phase (3)), we do not present the details here.

It is only when the answer to this phase is positive that it is useful to continue to the next phase, since causal queries that are not identifiable in the underlying single agent model will not be identifiable in a MACM representing the same JPD.

6.2. Negotiation

In this part it is first investigated whether each agents' model is extensive enough to be able to perform causal inference. More specifically, for each c-component in the underlying

single agent model, all its variables and immediate parents must be contained in the model of one of the agents. As we have seen in Lemma 3.6, this is because these are exactly the variables needed for calculating the c-factor of a c-component.

If the answer is negative, a negotiation between the agents ensues, with the goal to change the MACM to make the calculation of the causal effect possible. This is done by bargaining over the disclosure of information concerning some specific variables. If the negotiation succeeds the information is exchanged, the probability distributions are updated and the actual computation of $P(y|do(x))$ can start. If the negotiation fails, multi-agent causal inference is not possible in that MACM.

Due to space constraints we will not present the negotiation algorithm here, the interested reader is referred to [3]. In the next section, we treat the actual identification of causal effects in MACMs, the main contribution of this article.

7. Multi-agent identification

In this section, we start by adapting the single agent lemmas of Section 3.4.2 to the multi-agent context. Then we proceed to an algorithm for calculating $P(y|do(x))$ in a bi-agent causal model.

7.1. Multi-agent lemmas

Here the three lemmas that constitute the single agent identification algorithm of Section 3.4.3 will be transformed into multi-agent lemmas. In this section, we will consider a bi-agent causal model (M_1, M_2), where $M_i = \langle V_{M_i}, G_{M_i}, P(V_{M_i}), K_i \rangle$ for $i \in \{1,2\}$.

Furthermore, let a topological order over the union of the variables $\cup V$ in both agents be $V_1 < \cdots < V_n$ and let $V^{(i)} = \{V_1, \ldots, V_i\}$ and $V^{(0)} = \emptyset$.

7.1.1. Calculating c-factors

In Section 3.4 we have seen that Tian's single agent identification algorithm for semi-Markovian causal models is based on a factorization of the JPD in c-factors of c-components. A method for computing c-factors in bi-agent causal models is introduced in the following lemma.

Lemma 7.1 (Bi-agent c-factor calculation). *If a c-component S and its immediate parents are contained in exactly one agent, i.e. if $S \cup Pa(S) \subset V_{M_i}$, then the c-factor $Q[S]$ can be calculated using only information contained in that particular agent as follows*:

$$Q[S] = \prod_{V_i \in S} P(v_i | pa(T_i) \setminus \{v_i\}) \tag{22}$$

where T_i is the c-component of $G_{V^{(i)}}$ that contains V_i.

Proof. This lemma is an application of Lemma 3.6, originally from [2]. Since $S_j \cup Pa(S_j) \subset V_{M_i}$, all the elements involved in Eq. (22) belong solely to agent-i and thus the calculation can be done by that agent alone. □

Remark that the topological order of variables V_{M_i} of one agent is not sufficient to calculate $Q[S]$, since it could be that variables in the other agent have an influence on the topological order of variables V_{M_i}. Therefore, the order over the union of the variables $\cup V$ is needed. This can easily be done by the agents exchanging their local views on the topological order of the intersection variables V_{M_1,M_2} and thus combining their knowledge without exchanging private information.

7.1.2. Shrinking c-factors

Here, we adapt single agent Lemma 3.6 for calculating the c-factor of a subset based on a bigger set to the multi-agent case. $G_{M_i,B}$ is the graph of agent-i reduced to the set B.

Lemma 7.2. *If for some $A \subset B \subset V_{M_i}$, A is an ancestral set in the graph $G_{M_i,B}$ $(A = Anc(A)_{G_{M_i,B}})$, then $Q[A]$ is identifiable from $Q[B]$ as follows*:

$$Q[A] = \sum_{V_i \in B \setminus A} Q[B] \tag{23}$$

Proof. As both A and B belong to the same agent, the ancestral set of A in the graph reduced to B $(Anc(A)_{G_{M_i,B}})$, will also belong to that same agent. Therefore, we can apply single agent Lemma 3.7 to obtain the result. □

This lemma implies that c-factors inside agents can be shrunken if they have an ancestral subset.

7.1.3. Sub-factorization

Here, we will treat a lemma for factorizing the c-factor over a set H into its c-components.

Lemma 7.3. *If $H \subset V_{M_i}$ and H is partitioned in c-components $H_1, \ldots, H_l$ in G_H, then*

$$Q[H] = \prod_{i=1}^{l} Q[H_i] \tag{24}$$

Furthermore, every $Q[H_i]$ is identifiable from $Q[H]$.

Proof. As $H \subset V_{M_i}$, the same proof as for the single agent case applies. See [2] for the details. □

7.2. Recursive shrinking of c-factors

In Function 1, we introduce a recursive function *Identify* for calculating $Q[C]$ from $Q[T]$ in general, if this is possible, by making use of Lemmas 7.1–7.3. If T is completely contained in one agent ($T \subset V_{M_i}$ in this case), the function respects the privacy of the two agents in the sense that it only uses variables from V_{M_i} in its computation.

Function 1 Bi-agent *Identify*$(C, T, Q[T])$

Input: sets of variables $C \subset T \subset V_{M_i}$.
Output: Expression for $Q[C]$ in terms of $Q[T]$ using only variables V_{M_i}.
Let $A = Anc(C)_{G_{M_i,T}}$ {=the ancestors of C in graph G_{M_i} restricted to variables T}
(1) IF $A = C$, output $Q[C] = \sum_{T \setminus C} Q[T]$
(2) IF $A = T$, return *#false*
(3) IF $C \subset A \subset T$
  (a) In G_A, C is contained in a c-component T'.
  (b) Compute $Q[T']$ from $Q[A] = \sum_{T \setminus A} Q[T]$ by Lemma 7.2
  (c) return *Identify*$(C, T', Q[T'])$

In the first case C is an ancestral set in T, and thus $Q[C]$ can be calculated from $Q[T]$ with the help of Lemma 7.2, as $T \subset V_{M_1}$. In the second case, when the ancestral set of C in T is equal to T itself, just as in the single agent case there is no way to calculate $Q[C]$ from $Q[T]$. In the final case, when the ancestral set A of C in T is a strict set in between C and T, we formulate a recursive call of *Identify*, where T is replaced by its subset T', which is the c-component of C in T. As $T \subset V_{M_i}$ the calculation of $Q[T']$ based on $Q[T]$ can happen in the same way as in the multi-agent case. Due to space requirements we refer to [2] for this specific aspect. *Identify* will always terminate in one of the two first cases after a finite number of steps.

7.3. Complete multi-agent identification algorithm

Here we present the complete algorithm for computing $P(y|do(x))$ in a bi-agent causal model, based on the lemmas and function introduced earlier in this section.

Consider a bi-agent causal model (M_1, M_2), as before. As stated before, X belongs to agent1 $(X \in V_{M_1})$ and Y belongs to agent2 $(Y \in V_{M_2})$. Then the underlying single agent semi-Markovian model has variables $\cup V = V_{M_1} \cup V_{M_2}$ and as graph $\cup G$ the union of the graphs G_{M_1} and G_{M_2}. We assume that each of the c-components $S_1, \ldots, S_k$ of $\cup V$ and their respective parents $Pa(S_i)$ are completely contained in either agent1 (V_{M_1}) or agent2 (V_{M_2}), or formally $\forall S_i : (S_i \cup Pa(S_i) \subset V_{M_1}) \vee (S_i \cup Pa(S_i) \subset V_{M_2})$. In the next paragraphs we will discuss the different steps of Algorithm 2.

The first step in the algorithm is the calculation of the c-factors of all c-components. Since all c-components and their parents are local to al least one agent this can be done for all c-components without needing private information from other agents, using Lemma 7.1. For c-components that have variables in the intersection, it is calculated in the agent containing that c-component and its parents. At the end of this phase we have calculated $Q[S_{M_i,k}]$, this is the c-factor of the kth c-component in agent-i, for all agents and all c-components. Also, since we assume X is contained in the model of agent1, the c-factor $Q[S_X]$ is calculated in that agent.

In the second step of the algorithm we search for the ancestors of Y. Since it is possible that the different agents contain ancestors of Y this must be done in a multi-agent setting. In [18] methods have been created to perform this task while only sharing information about shared variables. At this point the ancestors of Y that are part of agent1 (containing X) are also known, and thus the intersection between $Anc(Y)$ and S_X, called D_X can be retrieved by agent1 since S_X is completely in that agent.

In the third step we divide the reduced graph G_{D_X} into c-components $D_{X,j}$, $j = 1, \ldots, l$. Then for each of these c-components the *Identify* function is called.

Finally, the results of the *Identify* calls are combined with the calculations of the c-factors $Q[S_{M_i,k}]$ retrieved in the first step to form the expression for $P(y|do(x))$. This results in a closed form formula divided in two parts, one that can be calculated completely in agent2 (part of the equation on line (27)) and one in agent1 (part of the equation on line (26)) and a summation over the variables in the intersection (part of the equation on line (25)). After their respective summations, distributions are obtained that only consist of $X \cup K$ for agent1 and $Y \cup K$ for agent2, where K is the intersection between the two agents. Hence, the resulting distribution of agent2 can be sent to agent1 without disclosing information except the intersection and the variable Y that is being studied in this specific query. Agent1 multiplies this distribution with its own result and performs the summation over $D_1 \cap (D_2 \backslash Y)$ to obtain the final result.

Now we will prove Eqs. (25)–(27).

Algorithm 2 Bi-agent identification of $P(y|do(x))$

Input: variables X and Y.
Output: $P(y|do(x))$ if it is identifiable, otherwise #*false*.

(1) Calculate all c-factors $Q[S_{M_i,k}]$ separately in each agent-i $i \in 1,2$ using Lemma 7.1. The c-factor of the c-component that contains X is identified by $Q[S_X]$ and is contained in agent1. The c-factors of the variables in the intersection are calculated in the agent that contains their parents.
(2) Find $D_i = Anc(Y)_{G_{V_{M_i} \backslash \{X\}}}$ in each agent-i $i \in 1,2$ and $D_X = D_1 \cap S_X$. See [18] for methods to do this in a multi-agent manner.
(3) Let the c-components of G_{D_X} be $D_{X,j}$, $j = 1, \ldots, l$.

For each set $D_{X,j}$:
Call *Identify*$(D_X, S_X, Q[S_X])$ given in Function 1 to calculate $Q[D_{X,j}]$. If the function returns #*false*, then stop and output #*false*.
Otherwise, output:

$$P(y|do(x)) = \sum_{D_1 \cap (D_2 \backslash Y)} \tag{25}$$

$$\sum_{D_1 \backslash K} \left[\prod_j Q[D_{X,j}] \prod_k \sum_{S_{M_1,k} \backslash D_1} Q[S_{M_1,k}] \right] \tag{26}$$

$$\sum_{D_2 \backslash (K \cup Y)} \left[\prod_k \sum_{S_{M_2,k} \backslash D_2} Q[S_{M_2,k}] \right] \tag{27}$$

Proof. This equation is based on the equations in Section 3.4.3 for identification in single agent models, originally from [2].

$Q[D_{X,j}]$ and $Q[S_{M_1,k}]$ consist only of elements in agent1 and $D_2 \backslash (K \cup Y)$, consists only of elements in agent2, thus the summation over $D_2 \backslash (K \cup Y)$ can be brought to the front. Likewise, $Q[S_{M_2,k}]$ is calculated solely from elements in agent2 and $D_1 \backslash K$ is part of agent1's

model. Combining the summations over $D_1 \cap (D_2 \setminus Y)$, $D_1 \setminus K$, and $D_2 \setminus (K \cup Y)$ yields a summation over $D \setminus Y$:

$$P(y|do(x)) = \sum_{D \setminus Y} \prod_j Q[D_{X,j}] \prod_k \sum_{S_k \setminus D} Q[S_k] \tag{28}$$

This is the single agent formula from Section 3.4.3. □

So we have introduced an algorithm for the identification of $P(y|do(x))$ in a bi-agent causal model, where each agent combines confidential information stored in its local model with information concerning the intersection with the other agent and the variables being studied (in this case X and Y). In this algorithm no information concerning other variables than the intersection and the variables X and Y is being disclosed.

7.4. Example

If we apply our algorithm to the product decision model in Fig. 8, with $X = \textit{product pricing}$ and $Y = \textit{product decision}$, to calculate the effect manipulating the product pricing strategy would have on the decision that will be made on some products. Then, we get $S_X = \{X, X_4, X_8\}$ and $Q[S_X] = P(x|x_1, x_2)P(x_4|x, x_1, x_2)P(x_8|x, x_1, x_2, x_3, x_4)$.

$D_1 = Anc(Y)_{G_{V \setminus X}} = \{X_2, X_3, X_4, X_5, X_6, X_7, X_8, X_9\}$ and $D_X = \{X_4, X_8\}$, then in the call $Identify(\{X_4, X_8\}, S_X, Q[S_X])$, $A(= Anc(\{X_4, X_8\})_{G_{S_X}} = \{X_4, X_8\})$ will be equal to C and $Q[X_4, X_8] = \sum_X Q[S_X]$ will be returned.

Finally,

$$P(y|do(x)) = \sum_{X_7, X_8, X_9} \sum_{X_2, \ldots, X_6} \left[\sum_X Q[S_X] \prod_i \sum_{S_{M_1,k} \setminus D_1} Q[S_{M_1,k}] \right] \tag{29}$$

$$\sum_{X_{10}, X_{11}} \left[\prod_i \sum_{S_{M_2,k} \setminus D_2} Q[S_{M_2,k}] \right] \tag{30}$$

where the $Q[S_{M_i,k}]$ can be calculated using Lemma 7.1 as follows. For agent1:

$$S_{M_1,1} = \{X_1\}; \quad S_{M_1,2} = \{X_2\}; \quad S_{M_1,3} = \{X_3\}; \quad S_{M_1,4} = \{X_5\}$$
$$S_{M_1,5} = \{X_6\}; \quad S_{M_1,6} = \{X_7\}; \quad S_{M_1,7} = \{X_9\}$$
$$Q[X_1] = P(x_1)$$
$$Q[X_2] = P(x_2)$$
$$Q[X_3] = P(x_3)$$
$$Q[X_5] = P(x_5|x_2)$$
$$Q[X_6] = P(x_6|x_4, x_5)$$
$$Q[X_7] = P(x_7|x, x_6)$$
$$Q[X_9] = P(x_9|x_3)$$

For agent2:

$$S_{M_2,1} = \{x_{10}\}; \quad S_{M_2,2} = \{x_{11}\}; \quad S_{M_2,3} = \{y\}$$
$$Q[X_{10}] = P(X_{10})$$
$$Q[X_{11}] = P(X_{11}|X_9, X_{10})$$
$$Q[Y] = P(Y|X_7, X_8)$$

This example shows that in a MACM we can calculate the causal effect of one variable in one agent on a variable in another agent without exchanging confidential information.

8. Conclusion and future work

In this article, we have shown the importance of CBN and their application for decision support systems. The biggest advantage of CBNs over traditional probabilistic BNs is that they sometimes allow to perform causal inference. This calculation of the effect of an intervention was based on a new type of conditioning, namely conditioning by manipulation. We studied a state-of-the-art causal inference algorithm in a centralized setting that allows to calculate causal effects when certain structural conditions are met. This algorithm is based on a new factorization of the JPD and is a systematic approach to the calculation.

We have elaborated on the problems that can arise when working with a centralized approach and discussed how a decentralized multi-agent approach might overcome some of these problems. Specific interest has been directed to cooperative multi-agent systems in which some internal information of individual agents has to remain private.

Multi-agent causal models is a way to overcome some of the problems in a single-agent setting. These MACMs can be learned from data of experts as can traditional BNs and CBNs and allow for a distributed calculation of causal effects.

We extended a state-of-the-art single-agent causal inference mechanism for this multi-agent setting. We showed that our algorithm is able to calculate all causal effects that could be calculated as if the entire domain was modeled by a single agent. When some of the structural constraints of the multi-agent model are not met at first, we referred to previous work where a negotiation phase was developed that can determine a new structure in which the calculation is possible. We also indicated previous work introducing techniques allowing to check beforehand whether the calculation is possible, without disclosing any sensitive information.

The main contribution of this article is the introduction of causal models and inference to the multi-agent setting, a step that has been neglected in research so far. Possible applications of these techniques are abundant, for example: distributed decision support systems, cooperative fraud detection, parallel computing performance analysis, etc.

We stress that in a MACM, agents are assumed to be honest and to cooperate to solve a problem, without disclosing their private information. Disclosure of confidential information via inference based on the combination of multiple non-confidential query results is a well known problem in statistical databases [24]. Investigating whether the solutions to this problem proposed there can be incorporated in our approach would be valuable future work.

In [4] we have started to investigate identification algorithms with >2 agents. Future work would be to study this issue in more detail.

Developing a method for calculating $P(s|do(t))$, where both T and S are sets, would also be an interesting extension of this work.

References

[1] J. Pearl, Causality: Models, Reasoning and Inference, MIT Press, 2000.
[2] J. Tian, J. Pearl, On the identification of causal effects, Tech. Rep. (R-290-L), UCLA C.S. Lab, March 2002.
[3] S. Maes, Multi-agent causal models: inference and learning, Ph.D. Thesis, Vrije Universiteit Brussel, December 2005.
[4] S. Maes, S. Meganck, B. Manderick, Identification in chain multi-agent causal models, in: Proceedings of the Special Track on Uncertain Reasoning of FLAIRS 2005, 2005.
[5] J. Pearl, Probabilistic Reasoning in Intelligent Systems, Morgan Kaufman, 1988.
[6] S.J. Russell, P. Norvig (Eds.), Artificial Intelligence: A Modern Approach, Prentice Hall, 1995.
[7] P. Spirtes, C. Glymour, R. Scheines, Causation, Prediction and Search, MIT Press, 2000.
[8] M.I. Jordan (Ed.), Learning in Graphical Models, MIT Press, 1998.
[9] J. Tian, J. Pearl, A general identification condition for causal effects, in: Proceedings of the National Conference on Artificial Intelligence (AAAI), 2002.
[10] J. Tian, J. Pearl, On the testable implications of causal models with hidden variables, in: Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence (UAI), 2002.
[11] M. Wooldridge (Ed.), An Introduction to Multi-agent Systems, John Wiley and Sons Ltd., 2002.
[12] G. Weiss (Ed.), Multi-agent Systems – A Modern Approach to Distributed Artificial Intelligence, MIT Press, 1999.
[13] Y. Xiang, Probabilistic Reasoning in Multi-agent Systems: A Graphical Models Approach, Cambridge University Press, 2002.
[14] P. de Oude, B. Ottens, G. Pavlin, Information fusion with distributed probabilistic networks, in: Proceedings of the 23rd IASTED International Multi-Conference on Artificial Intelligence and Applications, 2005, pp. 195–201.
[15] G. Pavlin, M. Maris, J. Nunnink, An agent-based approach to distributed data and information fusion, in: Proceedings of the IEEE/WIC/ACM Joint Conference on Intelligent Agent Technology (IAT04), 2004, pp. 466–470.
[16] B. Chaib-draa, Cm-relview: A tool for causal reasoning in multi-agent environments, in: Proceedings of the IEEE/WIC/ACM Joint Conference on Intelligent Agent Technology (IAT01), 2001.
[17] D. Koller, A. Pfeffer, Object-oriented Bayesian networks, in: Proceedings of the 13th Annual Conference on Uncertainty in Artificial Intelligence (UAI-97), 1997, pp. 302–313.
[18] S. Maes, J. Reumers, B. Manderick, Identifiability of causal effects in a multi-agent causal model, in: Proceedings of the 2003 IEEE/WIC International Conference on Intelligent Agent Technology (IAT), 2003.
[19] S. Nadkarni, P.P. Shenoy, A Bayesian network approach to making inferences in causal maps, European Journal of Operational Research 128 (2001) 479–498.
[20] S. Meganck, S. Maes, P. Leray, B. Manderick, Distributed learning of multi-agent causal models, in: Accepted at the 2005 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT'05), 2005.
[21] S. Maes, S. Meganck, B. Manderick, Multi-agent identification of causal effects, in: C. Ghidini, P. Giorgini, W. van der Hoek (Eds.), Proceedings of the 2004 European Workshop on Multi-Agent Systems (EUMAS), 2004, pp. 409–417.
[22] S. Maes, S. Meganck, B. Manderick, Causal inference in multi-agent causal models, in: Proceedings of Modèles Graphiques Probabilistes pour la Modélisation des Connaissances, Atelier of EGC 05, 2005, pp. 53–62.
[23] S. Maes, S. Meganck, B. Manderick, Identification of causal effects in multi-agent causal models, in: Proceedings of the IASTED International Conference on Artificial Intelligence and Applications, 2005, pp. 178–182.
[24] C. Boyens, O. Gunther, H.-J. Lenz, Handbook of Computational Statistics, Springer-Verlag, 2004, pp. 267–292 (Chapter 9).