A Conditional Independence Algorithm for Learning Undirected Graphical Models

Christian Borgelt

European Center for Soft Computing
Campus Mieres, Edificio Científico-Tecnológico
c/ Gonzalo Gutiérrez Quirós s/n, 33600 Mieres, Asturias, Spain
email: christian.borgelt@softcomputing.es

Abstract

When it comes to learning graphical models from data, approaches based on conditional independence tests are among the most popular methods. Since Bayesian networks dominate research in this field, these methods usually refer to directed graphs, and thus have to determine not only the set of edges, but also their direction. At least for a certain kind of possibilistic graphical models, however, undirected graphs are a much more natural basis. Hence, in this area, algorithms for learning undirected graphs are desirable, especially, since first learning a directed graph and then transforming it into an undirected one wastes resources and computation time. In this paper I present a general algorithm for learning undirected graphical models, which is strongly inspired by the well-known Cheng–Bell–Liu algorithm for learning Bayesian networks from data. Its main advantage is that it needs fewer conditional independence tests, while it achieves results of comparable quality.

Key words: graphical models, possibilistic network, learning from data, conditional independence test

1 Introduction

After modeling complex domains with graphical models, especially Bayesian networks, had become a popular research area in the eighties [19,24,17,9,15,12], since it allowed for consistent dependence modeling and efficient reasoning, learning graphical models from data was a focus of attention in the nineties [4,21,7,14,5,16,23,6,3,18]. Several search strategies and scoring functions, which are the two core ingredients of any algorithm for learning graphical models, were developed and applied not only to learning probabilistic graphical models, but also for the somewhat less well known possibilistic networks [13,1,8,3].

Among such learning algorithms, methods that are based on conditional independence (CI) tests are highly popular [22,5,23,6], because they possess several advantages over other methods. For example, they do not restrict the structure of the graphs that can be learned and do not require any prior information about the domain (while the famous K2 algorithm, for example, requires, at least in its basic form, a topological order of the attributes as input). They may even be used to enhance other methods, since they can be used to find missing prior information (in [21], for example, conditional independence tests are used to determine the topological order needed by the K2 algorithm).

Unfortunately, though, conditional independence test approaches are usually developed for directed graphs. To some degree this is understandable, since the set of conditional independence statements that can be represented by undirected graphs is monotonic in the condition sets (that is, if a conditional independence statement holds, expanding its set of conditioning attributes must not invalidate the conditional independence). As a consequence, undirected graphs cannot express certain (in)dependence structures that are fairly common in practice. In particular, they cannot capture a structure in which two variables are independent unconditionally, but become dependent conditional on a third variable, which is typical for a situation in which an effect can be brought about—jointly or independently—by two causes. This restriction is also one of the main reasons for the predominance of Bayesian networks (which, since they are based on directed graphs, allow for capturing such dependence structures) in the research on graphical models.

However, at least for a certain kind of possibilistic networks, which are based on a specific interpretation of degrees of possibility [3], undirected graphs are a much more natural basis. Of course, if one desires an undirected graphical model, it is always possible to turn a found directed graph into a corresponding undirected one by simply "moralizing" the graph (that is, by adding edges between non-adjacent parents of a node). This may lose independence information, but cannot lead to incorrect models. Nevertheless it is desirable to have methods that can learn undirected graphical models directly. In such algorithms the missing edge directions can be exploited, for example, to avoid certain tests and thus to achieve faster and possibly also more robust learning.

In this paper I present an algorithm for learning undirected graphical models that is strongly inspired by the well-known Cheng–Bell–Liu algorithm for learning Bayesian networks [5,6]. Its main advantage is that it requires fewer conditional independence tests, while it achieves results of comparable quality. This paper is organized as follows: in Sections 2 and 3 I briefly review some basics of CI tests needed in the algorithm and in Section 4 recall the Cheng–Bell–Liu algorithm. In Section 5 I present my algorithm for learning undirected graphical models. Section 6 reports experiments which compare the results of the two algorithms and Section 7 draws conclusion from the discussion.

2 Conditional Independence and Graphical Models

In general, graphical models are based on the idea to exploit the structural similarity of the sets of conditional independence statements that can hold in high-dimensional (probability or possibility) distributions and the sets of (node) separation statements that can hold in (directed or undirected) graphs. To be more specific, both conditional independence statements and (node) separation statements satisfy the so-called *(semi-)graphoid axioms*:

Definition 1 *Let V be a set of (mathematical) objects and $(\cdot \perp\!\!\!\perp \cdot \mid \cdot)$ a three-place relation of subsets of V. Furthermore, let W, X, Y, and Z be four disjoint subsets of V. Then the four statements*

$$\begin{array}{ll}
\textit{symmetry:} & (X \perp\!\!\!\perp Y \mid Z) \;\Rightarrow\; (Y \perp\!\!\!\perp X \mid Z) \\
\textit{decomposition:} & (W \cup X \perp\!\!\!\perp Y \mid Z) \;\Rightarrow\; (W \perp\!\!\!\perp Y \mid Z) \wedge (X \perp\!\!\!\perp Y \mid Z) \\
\textit{weak union:} & (W \cup X \perp\!\!\!\perp Y \mid Z) \;\Rightarrow\; (X \perp\!\!\!\perp Y \mid Z \cup W) \\
\textit{contraction:} & (X \perp\!\!\!\perp Y \mid Z \cup W) \wedge (W \perp\!\!\!\perp Y \mid Z) \;\Rightarrow\; (W \cup X \perp\!\!\!\perp Y \mid Z)
\end{array}$$

are called the semi-graphoid axioms. *A three-place relation $(\cdot \perp\!\!\!\perp \cdot \mid \cdot)$ that satisfies the semi-graphoid axioms for all W, X, Y, and Z is called a* semi-graphoid. *The above four statements together with*

$$\textit{intersection:} \quad (W \perp\!\!\!\perp Y \mid Z \cup X) \wedge (X \perp\!\!\!\perp Y \mid Z \cup W) \;\Rightarrow\; (W \cup X \perp\!\!\!\perp Y \mid Z)$$

are called the graphoid axioms. *A three-place relation $(\cdot \perp\!\!\!\perp \cdot \mid \cdot)$ that satisfies the graphoid axioms for all W, X, Y, and Z is called a* graphoid.

The rationale underlying graphical models is that the three-place relation named in this definition may either be interpreted as conditional independence or as separation, thus making it possible to use graphs as a "language" for representing sets of conditional independence statements. If the three-place relation is interpreted as conditional independence, $X \perp\!\!\!\perp Y \mid Z$ means that

$$\begin{array}{l}
\forall x \in \mathrm{dom}(X) : \forall y \in \mathrm{dom}(Y) : \forall z \in \mathrm{dom}(Z) : \\
\quad P(X = x, Y = y \mid Z = z) = P(X = x \mid Z = z) \cdot P(Y = y \mid Z = z)
\end{array}$$

in the probabilistic case and

$$\begin{array}{l}
\forall x \in \mathrm{dom}(X) : \forall y \in \mathrm{dom}(Y) : \forall z \in \mathrm{dom}(Z) : \\
\quad \Pi(X = x, Y = y \mid Z = z) = \max\{\Pi(X = x \mid Z = z), \Pi(Y = y \mid Z = z)\}
\end{array}$$

in the possibilistic case. It is easy to show that the semi-graphoid axioms hold for both probabilistic and possibilistic conditional independence and that the graphoid axioms hold for strictly positive probability measures.

What $X \perp\!\!\!\perp Y \mid Z$ means if it is interpreted as a statement about separation (of nodes) in a graph depends on whether the graph is directed or undirected. If it is undirected, the meaning is simply the following:

Definition 2 *Let $G = (V, E)$ be an undirected graph and X, Y, and Z three disjoint subsets of nodes. Z* u-separates *X and Y in G, written $\langle X \mid Z \mid Y \rangle_G$, iff all paths from a node in X to a node in Y contain a node in Z. A path that contains a node in Z is called* blocked *(by Z), otherwise it is called* active.

For directed graphs the conditions are slightly more complicated:

Definition 3 *Let $\vec{G} = (V, \vec{E})$ be a directed acyclic graph and X, Y, and Z three disjoint subsets of nodes. Z* d-separates *X and Y in $\vec{G}$, written $\langle X \mid Z \mid Y \rangle_{\vec{G}}$, iff there is* no *path from a node in X to a node in Y along which the following two conditions hold:*

(1) every node with converging edges either is in Z or has a descendant in Z,
(2) every other node is not in Z.

A path satisfying the two conditions above is said to be active*, otherwise it is said to be* blocked *(by Z).*

It is easy to show that both u-separation and d-separation satisfy the graphoid axioms. Hence one may use an appropriate graph to capture the set of conditional independence statements that hold in a given distribution.

Definition 4 *Let $(\cdot \perp\!\!\!\perp_\delta \cdot \mid \cdot)$ be a three-place relation representing the set of conditional independence statements that hold in a given distribution δ over a set U of attributes. A (directed or undirected) graph $G = (U, E)$ over U is called a* conditional dependence graph *or a* dependence map *w.r.t. δ, iff for all disjoint subsets $X, Y, Z \subseteq U$ of attributes*

$$X \perp\!\!\!\perp_\delta Y \mid Z \;\Rightarrow\; \langle X \mid Z \mid Y \rangle_G,$$

i.e., if G captures by u-separation all (conditional) independences that hold in δ and thus represents only valid (conditional) dependences. Similarly, G is called a conditional independence graph *or an* independence map *w.r.t. δ, iff for all disjoint subsets $X, Y, Z \subseteq U$ of attributes*

$$\langle X \mid Z \mid Y \rangle_G \;\Rightarrow\; X \perp\!\!\!\perp_\delta Y \mid Z,$$

i.e., if G captures by u-separation only (conditional) independences that are valid in δ. G is said to be a perfect map *of the conditional (in)dependences in δ, if it is both a dependence map and an independence map.*

Although the correspondence cannot always be made perfect, it is a very convenient tool. Together with the core theorem of graphical models, which connects conditional independence graphs with decompositions of distributions (factorizations in the case of probability distributions), these definitions are the basis of using graphs to capture essential properties of distributions and to derive consistent and efficient methods for drawing inferences in them. W.r.t. learning graphical models from data, these definitions provide the basis for induction algorithms based on conditional independence tests.

3 Conditional Independence Tests

The core ingredient of the algorithms for learning graphical models, which I focus on in this paper, is a conditional independence test (CI test for short). Such a test usually consists of a scoring function for assessing the strength of the (conditional) dependence of two variables together with a threshold. If the value of the scoring function is below the threshold, the variables are considered to be (conditionally) independent, otherwise they are judged to be (conditionally) dependent. By checking which conditional independences hold, one tries to find a suitable conditional independence graph. Which conditional independences are tested (it is immediately clear that we cannot perform an exhaustive search) is determined by the search strategy (see Sections 4 and 5).

There is an abundance of possible scoring functions, especially for (conditional) probabilistic (in)dependence. The reason is that apart from standard dependence measures used in classical statistics, basically all measures used in decision tree induction can be transferred and then yield such a scoring function. Even if a measure was originally only intended for assessing the strength of marginal dependence, it can usually be extended to yield a measure for conditional dependence by simply computing its value for each possible instantiation of the conditioning attributes and then aggregating these values in a suitable manner. For example, one of the most common measures for the strength of marginal (probabilistic) dependence is *information gain*,

$$I_{\text{gain}}(A,B) = \sum_{i,j} p_{ij} \log_2 \frac{p_{ij}}{p_{i.}p_{.j}} = -\sum_i p_{i.} \log_2 p_{i.} + \sum_j p_{j.} \sum_i p_{i|j} \log_2 p_{i|j}$$

(where A and B are two variables, i and j range over their respective values, p_{ij}, $p_{i.}$ and $p_{.j}$ are the probabilities of the joint and individual occurrence of these values, and conditional probabilities are defined in the usual way). It can be extended to conditional information gain as

$$I_{\text{gain}}(A,B|C) = \sum_k p_{..k} \sum_{i,j} p_{ij|k} \log_2 \frac{p_{ij|k}}{p_{i.|k}p_{.j|k}},$$

that is, by simply summing its value over all possible instantiations of the conditions. (Note that here C may be an individual variable or a set of variables; and then k refers to individual values or value vectors, respectively.)

For the transfer to the possibilistic case, which is discussed below, it is relevant that information gain may be written conveniently as

$$I_{\text{gain}}(A, B) = H(p_A) + H(p_B) - H(p_{AB}),$$

where $H(p) = -\sum_i p_i \log_2 p_i$ denotes the well-known *Shannon entropy* of the probability distribution p.

In order to remove the tendency of information gain to over-assess the strength of dependence of many-valued attributes (which is not a fundamental property, but simply results from the discretization of the—estimated—probability values due to the necessarily limited size of a database to learn from; see [3]), it is often normalized to *information gain ratio*

$$I_{\text{gr}}(A, B) = \frac{I_{\text{gain}}(A, B)}{H(p_A)} = \frac{\sum_{i,j} p_{ij} \log_2 \frac{p_{ij}}{p_{i.}p_{.j}}}{-\sum_i p_i \log_2 p_i}$$

or one of its two *symmetrical forms*

$$I_{\text{sgr1}}(A, B) = \frac{I_{\text{gain}}(A, B)}{H(p_A) + H(p_B)} = \frac{\sum_{i,j} p_{ij} \log_2 \frac{p_{ij}}{p_{i.}p_{.j}}}{-\sum_i p_i \log_2 p_i - \sum_j p_j \log_2 p_j} \qquad \text{and}$$

$$I_{\text{sgr2}}(A, B) = \frac{I_{\text{gain}}(A, B)}{H(p_{AB})} = \frac{\sum_{i,j} p_{ij} \log_2 \frac{p_{ij}}{p_{i.}p_{.j}}}{-\sum_{ij} p_{ij} \log_2 p_{ij}}.$$

Worth mentioning is, of course, also the well-known χ^2 measure,

$$\chi^2(A, B) = N_{..} \sum_{i,j} \frac{(p_{i.}p_{.j} - p_{ij})^2}{p_{i.}p_{.j}},$$

where $N_{..}$ is the total number of sample cases in the database to learn from. Extended to conditional tests it reads

$$\chi^2(A, B|C) = N_{..} \sum_k p_{..k} \sum_{i,j} \frac{(p_{i.|k}p_{.j|k} - p_{ij|k})^2}{p_{i.|k}p_{.j|k}}.$$

Other probabilistic dependence measures include, for example, the Gini index or the relief measure (see [3] for details and a more extensive list of measures).

Not surprisingly, there is also a variety of measures for the possibilistic case [1,3]. A fairly well-known example is the so-called *specificity gain*,

$$S_{\text{gain}}(A, B) = \text{nsp}(\pi_A) + \text{nsp}(\pi_B) - \text{nsp}(\pi_{AB}).$$

It is easy to see that it is formed in direct analogy to information gain, with the Shannon entropy of a probability distribution replaced by the *nonspecificity* $\text{nsp}(\pi)$ of a possibility distribution π. This nonspecificity is defined as

$$\text{nsp}(\pi) = \int_0^{\sup_{E\in\mathcal{E}} \pi(E)} \log_2 \Big(\sum_{E\in\mathcal{E}} [\pi]_\alpha(E) \Big)\, d\alpha,$$

where $\mathcal{E}$ is the set of elementary events underlying the possibility distribution π (that is, the set of distinct instantiations of the variable or variables) and $[\pi]_\alpha$ denotes the indicator function of the α-cut of the possibility distribution π (that is, $[\pi]_\alpha(E) = 1$ if $\pi(E) \geq \alpha$ and $[\pi]_\alpha(E) = 0$ otherwise).

Other measures include the following, which I used to obtain the experimental results reported in Section 6: in the first place, specificity gain may be normalized in analogy to information gain, which serves the purpose to eliminate or at least reduce a possible tendency to overrate the strength of dependence of many-valued attributes. This leads to the *specificity gain ratio*,

$$S_{\text{gr}}(A, B) = \frac{S_{\text{gain}}(A, B)}{\text{nsp}(\pi_A)} = \frac{\text{nsp}(\pi_A) + \text{nsp}(\pi_B) - \text{nsp}(\pi_{AB})}{\text{nsp}(\pi_A)}$$

and two *symmetric specificity gain ratios*, namely

$$S_{\text{sgr1}}(A, B) = \frac{S_{\text{gain}}(A, B)}{\text{nsp}(\pi_{AB})} \qquad \text{and} \qquad S_{\text{sgr2}}(A, B) = \frac{S_{\text{gain}}(A, B)}{\text{nsp}(\pi_A) + \text{nsp}(\pi_B)}.$$

By exploiting the fact that information gain can be written in two ways, another possibilistic measure can be derived. It relies on the form that is usually called "mutual information" (even though it is the same measure as information gain, since the formulas are equivalent) and compares the actual joint distribution and a hypothetical independent distribution by computing and summing their pointwise quotients, that is,

$$d_{\text{mi}}(A, B) = -\sum_{i,j} \pi_{ij} \log_2 \frac{\pi_{ij}}{\min\{\pi_{i.}, \pi_{.j}\}},$$

where π_{ij} denotes the degree of possibility of the combination of the i-th value of attribute A and the j-th value of attribute B. Furthermore, $\pi_{i.}$ and

$\pi_{j.}$ denote the marginal degrees of possibility, which result from taking the maximum over all values of the missing index.

Finally, the χ^2 measure (see above), which computes and sums pointwise squared differences between the actual joint distribution and a hypothetical independent distribution, can be transferred to possibility distributions. Thus we obtain the following measure:

$$d_{\chi^2}(A,B) = \sum_{i,j} \frac{(\min\{\pi_{i.}, \pi_{.j}\} - \pi_{ij})^2}{\min\{\pi_{i.}, \pi_{.j}\}}.$$

Other scoring functions, as well as a more extensive discussion of the ideas underlying them, can be found in [3].

All of these (possibilistic) scoring functions can be extended to conditional tests by summing their values over all instantiations of the conditioning attributes, weighted with the degree of possibility of the instantiation. This mimics the way in which conditional measures are constructed in the probabilistic case from their marginal counterparts (see above).

A general problem of conditional independence tests is their order, that is, the number of conditioning attributes. Since the data sets that are available in practice are always of limited size, conditional independence tests become quickly unreliable, the higher their order. Hence algorithms for learning graphical models from data must take care to keep the order of conditional independence tests under control. For example, the Cheng–Bell–Liu algorithm, which is reviewed in the next section, does so by exploiting an already constructed graphical model to determine suitable condition sets. If the occurring condition sets still become too large, it is usually fixed that all conditional independence tests with an order higher than a user-specified maximum fail.

4 The Cheng–Bell–Liu Algorithm

The Cheng–Bell–Liu algorithm [5,6] constructs a Bayesian network from a data set of sample cases. It assumes that the given domain can accurately be modeled by a perfect map, that is, by a directed graph that captures exactly the conditional independences that are present in the probability distribution governing the data generation process. In addition, it assumes that the scoring function used for the conditional independence tests is, in a certain sense, well-behaved. Among other things, this means that the result of a conditional independence test coincides with the actual situation (that is, the test succeeds if and only if the tested pair of attributes is actually (conditionally) independent). In addition, if a condition that is needed for the conditional

independence of two attributes is removed, the value of the measure should increase, regardless of what other attributes are present in the conditions. For detailed discussion of the exact assumptions and preconditions underlying the Cheng–Bell–Liu algorithm, see [6].

The Cheng–Bell–Liu algorithm works in four phases:

4.1 Drafting

A so-called Chow–Liu tree [10] is built as an initial graphical model. This involves evaluating all possible edges (that is, pairs of attributes) with an (unconditional) independence test. The standard form of the algorithm uses information gain as the scoring function and an independence threshold of about 0.1 bits. Edges (identified by attribute pairs) having a score lower than this threshold are permanently discarded from the construction process, since the incident attributes are considered to be (marginally) independent. The rest form a set of candidate edges, which are weighted with the value of the scoring function. The initial graphical model is then formed by constructing an optimum weight spanning tree with these edge weights.

4.2 Thickening

The candidate edges that were not used for the initial graphical model (that is, are not contained in the constructed spanning tree) are traversed in the order of decreasing score. For each of these edges it is tested whether it may be needed in the graphical model. The test exploits the current graphical model in order to find a good set of conditions, namely by selecting the set of adjacent nodes of one attribute that lie on paths leading to the other attribute.

The rationale underlying this scheme is the so-called *local Markov property* of directed conditional independence graphs: an attribute is conditionally independent of all non-descendants given its parents (cf. the notion of d-separation, defined in Definition 3, which implies this property in a conditional independence graph). Of course, since the graph is undirected at this point we cannot determine which attribute is a non-descendant of the other, and hence both possibilities may have to be tried (one trial with the neighbors of one attribute, another with the neighbors of the other). In addition, the set of adjacent nodes may not only include parents, but also children. Children are a problem, because in the true graphical model there may be a so-called *v-structure* on the path between the nodes (that is, there may be a node at which edges converge). Including such a node (which may be a child) in the conditions is harmful as it activates the path and thus hinders conditional independence.

Therefore the set of adjacent nodes is reduced iteratively and greedily. In each step the conditioning attribute, which, if removed, lowers the dependence score the most, is discarded. This reduction is carried out until the score falls below the given independence threshold or no removal of a conditioning attribute lowers the score. In the former case, if it occurs in either of the the two trials, the attributes of the tested edge are judged to be conditionally independent and the edge is not added to the graphical model. Otherwise it is added.

4.3 Thinning

In the thickening phase a test whether an edge is needed was based on a graph that was possibly still to sparse to reveal the conditional independence of a pair of attributes. This is the case, because not all paths that exist between the two attributes in the true model may have been present at the time of the test (even though they must all be present at the end of the thickening phase, provided the assumptions underlying the algorithm hold [6]). Hence the edges that are present in the graphical model after the thickening phase are traversed again and it is retested whether they are needed.

This test is carried out in two ways: first in the way described on the thickening phase (which is the so-called "heuristic" form of the test). However, there are certain degenerate cases where this test does not correctly identify an existing conditional independence and thus a superfluous edge may not be removed (see [6] for an example). Hence a strict test, which adds the neighbors of the neighbors of an attribute to the condition set (because two adjacent nodes on a path cannot both have v-structures) before this set is reduced, is carried out (this is called the so-called "strict" form of the test).

If any of these tests reveals that two attributes are conditionally independent, the corresponding edge is removed from the graph. It can be shown that the resulting graph is a skeleton for the perfect map describing the domain (that is, it contains exactly the needed edges, only directions are missing)—provided, of course, the underlying assumptions hold.

4.4 Orienting

In the last phase the edges of the graph are directed. This is done in two steps: first the v-structures (converging edges) are identified and then the remaining edges are oriented, using rules that avoid the introduction of v-structures and cycles. (The latter step may be done fully automatically or may be supported manually, since the v-structures usually do not uniquely determine the direction of all edges and thus human intervention may be advisable.)

In order to find the v-structures, all pairs of attributes with common neighbors are traversed and it is determined by a (strict) conditional independence test (as described above), which common neighbors are in the (maximally) reduced set of conditions that render the attributes conditionally independent. Those attributes that are removed from the condition set must be common children (since they do not contribute to rendering the attributes conditionally independent, while fixing parents is essential for achieving conditional independence due to the local Markov property). Hence a v-structure is built with them (that is, the edges are directed towards the common neighbor).

Afterwards other edges may be directed by alternatingly extending directed chains and choosing random directions for edges, the direction of which is not fixed by the v-structures and already extended chains. The rationale underlying this procedure is that extending existing chains is the only way to avoid additional v-structures. Furthermore, cycles must, of course, be avoided.

5 Learning Undirected Graphs

It is obvious that one way of constructing an undirected graphical model consists in executing the Cheng–Bell–Liu algorithm, which yields a directed graph, and then moralizing the result (that is, adding edges between non-adjacent parents of a node). However, from the above description of the algorithm it is clear that in this case a lot of unnecessary work is done. For example, edges are directed and then their direction is discarded again when the graph is moralized. In the tests whether an edge is needed, it is tried to remove children from the condition sets even though in an undirected graph there is no concept of child or parent. Finally, the "strict" form of a conditional independence test has to add neighbors of neighbors of the attributes in order to make sure that all paths are securely blocked, regardless of any possibly existing v-structures, thus introducing a strong tendency towards conditional independence tests of fairly high order. Hence it is desirable to devise an algorithm that removes this unnecessary work and thus obtains an undirected graphical model faster and possibly also in a more reliable way.

The algorithm I propose for this task works in four phases:

5.1 Drafting

This phase is identical to the Cheng–Bell–Liu algorithm, that is, a Chow–Liu tree is formed (that is, an optimum weight spanning tree is constructed from edge weights that represent dependence strengths).

5.2 Thickening

As in the Cheng–Bell–Liu algorithm, the remaining candidate edges (that is, edges with a score above the threshold, but not used in the initial graphical model) are traversed and tested. If the test indicates that they may be needed (because the incident attributes are not conditionally independent given the chosen conditions), they are added to the graphical model.

The difference to the Cheng–Bell–Liu algorithm consists in how the conditional independence tests are executed. The underlying principle is analogous, but exploits the *local Markov property* of *undirected* graphs: an attribute is conditionally independent of any other attribute given its neighbors (cf. the notion of u-separation, defined in Definition 2, which implies this property in a conditional independence graph). We may even restrict the set of neighbors to those lying on paths leading to the other attribute (even though this is, strictly speaking, already part of the so-called *global Markov property*).

Note that there is no need for any (greedy or non-greedy) reduction of the condition set, since we need not take care of children with v-structures. In principle, only a single conditional independence test is needed per edge. However, in order to improve the robustness, but also the efficiency of the algorithm, it may be advisable to carry out the alternative test (that is, using the neighbors of the other attribute) if it fails. The reason is that a test may fail, because the current graph is still too sparse and thus not all neighbors that are necessary to render the attributes conditionally independent are already present. If this is the case for one attribute, there is still a chance that the set of neighbors of the other attribute is already complete and thus this test is worthwhile.

Clearly, if the executed test indicates that the attributes are *not* conditionally independent given the current graph structure, the edge is added to the graph. Otherwise it is discarded.

5.3 Moralizing

If one assumes that there exists an undirected perfect map of the domain under consideration, the thinning phase (see below) already yields the result. However, this would not be a feasible assumption. Dependence structures that contain (directed) v-structures are much too frequent in practice and unfortunately there is no lossless way of representing v-structures in undirected graphs. Hence simply assuming that there is an undirected perfect map would render the algorithm basically useless for practical purposes.

However, in order to take care of v-structures one only has to connect the par-

ents, that is, one has to moralize the graph. The reason is that even though the parents are independent given their common ancestors (which may be the empty set), they become dependent once a common child is given. This non-monotone behavior (enlarging the set of conditions destroys a conditional independence) cannot be expressed in undirected graphs (cf. Section 2). Thus one must allow for unrepresented conditional independences. However, this is not too much of a problem, since at least for reasoning purposes an independence map suffices—we do not necessarily need a perfect map.

The consequences of these considerations for the algorithm are clear: some of the edges, which have an unconditional score below the threshold (and thus were discarded before the initial graphical model was constructed) or were discarded in the thickening phase, may be needed in the graphical model in order to achieve monotony w.r.t. conditional independence. However, for these tests only edges need to be considered that have a common neighbor in the graph constructed so far, since no other edges can connect nodes that are involved in a v-structure. Therefore all such edges are traversed and it is tested whether they are needed in the graph. If the corresponding attributes are *not* found to be conditionally independent given the current graph (that is, given all neighbors of one of the attributes), the edge is added.

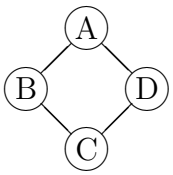
5.4 Thinning

As for the Cheng–Bell–Liu algorithm it holds that the graph resulting from the preceding step may contain superfluous edges, since when the test was carried out not all necessary edges and paths may have been present in the graph. Hence all edges of the graph are retested and those found to be unnecessary are removed from the graph. (Note that both edges added in the thickening step as well as edges added in the moralizing step are retested.)

5.5 Additional Thinning

As a simple extension of this algorithm one may add a second thinning phase between the thickening phase and the moralizing phase. The idea of such a phase is that the graph resulting from the thinning phase may contain fewer attribute pairs with common neighbors, so that fewer tests have to be carried out in the moralizing phase. Furthermore, the order of the conditional independence tests may be lower, because a lower number of edges can generally be expected to reduce the number of neighbors that enter the condition sets.

The additional costs for such a phase are negligible, since the results of already executed conditional independence tests will be stored in a cache anyway (to



p_{ABCD}		$A = a_1$		$A = a_2$	
		$B = b_1$	$B = b_2$	$B = b_1$	$B = b_2$
$C = c_1$	$D = d_1$	$^{16}/_{82}$	$^{4}/_{82}$	$^{4}/_{82}$	$^{4}/_{82}$
	$D = d_2$	$^{4}/_{82}$	$^{1}/_{82}$	$^{4}/_{82}$	$^{4}/_{82}$
$C = c_2$	$D = d_1$	$^{4}/_{82}$	$^{4}/_{82}$	$^{1}/_{82}$	$^{4}/_{82}$
	$D = d_2$	$^{4}/_{82}$	$^{4}/_{82}$	$^{4}/_{82}$	$^{16}/_{82}$

Fig. 1. Example graphical model that can be learned with the proposed algorithm, but not with the Cheng–Bell–Liu algorithm.

avoid redundant computations). Hence only for edges between attributes that received a new incident edge in the moralizing step a new test has to be carried out in the second thinning phase. The result of all other tests are already present in the cache and thus can be reused basically without costs.

5.6 Illustrative Remarks

In order to illustrate the difference of the proposed algorithm to the Cheng–Bell–Liu algorithm, in particular w.r.t. their ability to learn undirected graphical models, consider the example shown in Figure 1. The (undirected) graph shown on the left is a perfect map of the probability distribution shown on the right. However, the Cheng–Bell–Liu algorithm is unable to discover this graph. The reason is that it makes the assumption that there is a directed perfect map of the domain under consideration. However, there is no directed perfect map for this probability distribution.

Nevertheless, carrying out the Cheng–Bell–Liu algorithm for this probability distribution (or a corresponding data set), and then moralizing the resulting graph, yields a proper undirected independence map of this probability distribution. Unfortunately, though, this map contains a superfluous edge, namely either the edge A—C or the edge B—D (which edges is selected depends on the order in which edges with the same score are processed). Note that the thinning phase of the Cheng–Bell–Liu algorithm cannot remove this edge, because it is actually introduced by the moralizing step and not part of the result of the Cheng–Bell–Liu algorithm (which contains an (incorrect) v-structure).

This example also reveals that using the Cheng–Bell–Liu algorithm to learn an undirected graphical model may fail to find the best undirected graphical model even if there exists an undirected perfect map of the domain and thus provides an additional argument in favor of the proposed algorithm. This algorithm finds the optimal model easily, since it is already the result of the thickening phase, and neither moralizing nor thinning change the graph.

6 Experiments

In order to test the learning algorithm proposed above, I implemented it as part of the INeS program (Induction of Network Structures), which also comprises a large number of other learning algorithms for probabilistic and possibilistic graphical models, including the Cheng–Bell–Liu algorithm. [1]

As a test case I chose a standard benchmark, namely the well-known BOBLO (BOvine BLOod) network and data set [20], which is also known as the Danish Jersey cattle data. It consists of a manually constructed Bayesian network over 21 attributes, which describe genotypical and phenotypical properties of dam, sire and calf, together with switch variables that specify whether dam or sire were correctly identified. The network serves the serves the purpose to verify parentage for pedigree registration. In addition, there is a real-world data set of 500 cases, a large number of which contain missing values.

I focused on learning a possibilistic network, since at least the kind, in which conditional distributions are identified with joint distributions on the same subspace, is more naturally represented with the help of an undirected graph. In the implementation I made use of a method to efficiently compute maximum projections of a multivariate possibility distribution [2], which is represented by a database of sample cases with missing values or set-valued information.

As already pointed out in Section 3, I collected scoring functions from [1,3], taking the restriction into account that the measure must be symmetric in order to be usable for a conditional independence test. As a consequence I chose the specificity gain S_{gain}, a symmetric ratio of the specificity gain S_{sgr1} (the two variants mentioned in Section 3 behave similarly), the possibilistic analog of the χ^2 measure d_{χ^2}, and the possibilistic analog of mutual information d_{mi} (see Section 3 for the definitions). As independence thresholds I used 0.015 for S_{gain}, 0.1 for S_{sgr1} and d_{χ^2}, and 0.04 for d_{mi}. These values were chosen, because they yielded reasonably good results. By tuning these values, the complexity of the learned network may be controlled to some degree: the higher the threshold, the sparser the learned network.

The results of the experiments together with information about the number of conditional independence tests needed are shown in Tables 1 and 2. Table 1 shows the size and the quality of the networks learned with the Cheng–Bell–Liu algorithm and subsequent moralization using different scoring functions.

[1] Note, however, that [5,6] do not specify all relevant details of the algorithm. Especially, there is no clear description how cycles introduced by v-structures—a situation that can occur due to certain numerical properties of information gain in connection with a threshold—are treated. Hence INeS may produce results that differ from the reference implementation of this algorithm.

Table 1
Network quality with Cheng–Bell–Liu Algorithm (and moralization)

measure	edges	params	min.	avg.	max.
S_{gain}	16	678	8.784	8.932	10.620
S_{sgr1}	20	4701	8.638	8.826	10.340
d_{χ^2}	14	9450	9.344	9.446	10.728
d_{mi}	17	1010	8.460	8.538	10.406

Table 2
Network quality of learning an undirected graphical model

measure	edges	params	min.	avg.	max.
S_{gain}	17	639	8.738	8.888	10.614
S_{sgr1}	21	3847	8.740	8.926	10.464
d_{χ^2}	12	3442	9.352	9.453	10.730
d_{mi}	19	570	8.586	8.656	10.558

The number of edges (in the moralized graph) and the number of parameters show the considerably different behavior of the measures w.r.t. the complexity of the learned network. The last three columns state the sums of the minimum, average, and maximum possibility degree of points covered by the tuples in the database, which should be as low as possible (see [3] for more details about this way of evaluating learned possibilistic graphical models).

As could also be observed in experiments with other learning algorithms, d_{χ^2} tends to learn large networks. However, they are usually of better quality than the ones built by the Cheng–Bell–Liu algorithm. This could not be improved by increasing the independence threshold: although the networks get smaller when doing so, they also get even worse, thus indicating that this measure may not be well suited for an approach based on conditional independence tests. With the possibilistic analog of mutual information d_{mi} and the specificity gain S_{gain}, however, results are obtained that are competitive with other learning approaches. This confirms earlier results (see [3]) that d_{mi} is among the most recommendable measures for learning possibilistic graphical models.

Table 2 shows the results for the proposed algorithm that learns undirected graphical models directly. It can be observed that it tends to learn smaller networks: even though the number of edges is the same or even slightly larger, the number of parameters is clearly smaller for all measures. Nevertheless the quality of the learned networks is comparable, only for d_{mi} the quality deteriorates slightly. However, this may be counteracted by adapting the independence threshold (a smaller threshold improves the network quality).

Table 3
Number of tests with strict Cheng–Bell–Liu Algorithm

	marg.	order of cond. tests					
measure	tests	1	2	3	4	5	any
S_{gain}	210	24	14	9	2	0	49
S_{sgr1}	210	67	50	33	20	3	173
d_{χ^2}	210	19	18	35	12	3	87
d_{mi}	210	28	37	26	9	0	100

Table 4
Number of tests with heuristic Cheng–Bell–Liu Algorithm

	marg.	order of cond. tests					
measure	tests	1	2	3	4	5	any
S_{gain}	210	24	14	9	2	0	49
S_{sgr1}	210	71	48	30	20	3	172
d_{χ^2}	210	19	18	32	11	3	83
d_{mi}	210	31	39	14	2	0	86

Table 5
Number of tests with algorithm for undirected graphs

	marg.	order of cond. tests					
measure	tests	1	2	3	4	5	any
S_{gain}	210	13	11	5	4	0	33
S_{sgr1}	210	42	29	16	3	0	90
d_{χ^2}	210	9	7	2	8	0	26
d_{mi}	210	13	15	20	0	0	48

My main goal when developing the proposed algorithm was to reduce the number of the conditional independence tests, especially of higher order. Therefore Tables 3 to 5 show the number of (conditional) independence tests that were carried out by the two algorithms. Since the drafting phase is identical in both algorithms, they execute the same number of marginal independence test, namely 210 ($= \frac{21 \cdot 20}{2}$). This number obviously cannot be reduced, since all edges must be tested marginally. However, there are considerable differences in the number of conditional independence tests carried out.

In order to treat the original Cheng–Bell–Liu algorithm as fairly as possible, I executed it in two versions. The first version is the one as it was described

above and the numbers of tests needed are shown in Table 3. The second version is simplified in as far as in the thinning phase it only carries out heuristic tests (using only direct neighbors in the condition sets) and skips the strict tests (using also neighbors of neighbors). Both versions lead to the same networks on this example problem and thus Table 1 applies for both version. However, in terms of the number of conditional independence tests one can expect fewer and lower order tests with the second version. This is confirmed by Table 4, although the gains are a lot smaller than I had expected. The reason may be that for the orienting phase strict tests are needed anyway and thus skipping them in the thinning phase does not help much. The biggest savings result for d_{mi}, where the number of tests goes down by 14.

In contrast to these fairly small gains the gains achieved by using the proposed algorithm for learning undirected graphs directly are considerable. The number of conditional independence tests is cut to about half of the tests needed by the Cheng–Bell–Liu algorithm. The biggest gains result for d_{χ^2}, but also d_{mi}, one of the most recommendable measure, benefits significantly. It is worth noting that for all measures except S_{gain} the highest order of a conditional independence test drops by 1.

7 Conclusions

In this paper I presented an algorithm for learning undirected graphical models from data that is based on conditional independence tests and inspired by the well-known Cheng–Bell–Liu algorithm. It is particularly useful for possibilistic networks, since at least for a certain kind of these networks undirected graphs are the most natural representation. The expectation that the algorithm would achieve its results with fewer conditional independence tests or conditional independence tests of lower order was clearly confirmed in the reported experiments. Hence if undirected graphical models are to be learned, the presented algorithm provides significant advantages. It may be conjectured that it is not only faster, but also provides more reliable results, since the number and order of the tests is decisive in this respect. However, this needs to be substantiated with more experiments.

Software

The INeS program, which is written in C and was used to carry out the experiments described above, can be downloaded free of charge at

`http://www.borgelt.net/ines.html`

References

[1] C. Borgelt and R. Kruse. Evaluation Measures for Learning Probabilistic and Possibilistic Networks. *Proc. 6th IEEE Int. Conf. on Fuzzy Systems (FUZZ-IEEE'97, Barcelona, Spain)*, Vol. 2:1034–1038. IEEE Press, Piscataway, NJ, USA 1997

[2] C. Borgelt and R. Kruse. Efficient Maximum Projection of Database-Induced Multivariate Possibility Distributions. *Proc. 7th IEEE Int. Conf. on Fuzzy Systems (FUZZ-IEEE'98, Anchorage, Alaska, USA)*, CD-ROM. IEEE Press, Piscataway, NJ, USA 1998

[3] C. Borgelt and R. Kruse. *Graphical Models — Methods for Data Analysis and Mining.* J. Wiley & Sons, Chichester, United Kingdom 2002

[4] G.F. Cooper and E. Herskovits. A Bayesian Method for the Induction of Probabilistic Networks from Data. *Machine Learning* 9:309–347. Kluwer, Dordrecht, Netherlands 1992

[5] J. Cheng, D.A. Bell, and W. Liu. Learning Belief Networks from Data: An Information Theory Based Approach. *Proc. 6th ACM Int. Conf. Information Theory and Knowledge Management (CIKM'97, Las Vegas, NV)*, 325–331. ACM Press, New York, NY, USA 1997

[6] J. Cheng, R. Greiner, J. Kelly, D.A. Bell, and W. Liu. Learning Bayesian Networks from Data: An Information Theory Based Approach. *Artificial Intelligence* 137(1–2):43–90. Elsevier, Amsterdam, Netherlands 2002

[7] W. Buntine. Operations for Learning with Graphical Models. *Journal of Artificial Intelligence Research* 2:159–225. Morgan Kaufman, San Mateo, CA, USA 1994

[8] L.M. de Campos, J.F. Huete, and S. Moral. Independence in Uncertainty Theories and Its Application to Learning Belief Networks. In: [11], 391–434.

[9] E. Castillo, J.M. Gutierrez, and A.S. Hadi. *Expert Systems and Probabilistic Network Models.* Springer, New York, NY, USA 1997

[10] C.K. Chow and C.N. Liu. Approximating Discrete Probability Distributions with Dependence Trees. *IEEE Trans. on Information Theory* 14(3):462–467. IEEE Press, Piscataway, NJ, USA 1968

[11] D. Gabbay and R. Kruse, eds. *DRUMS Handbook on Abduction and Learning.* Kluwer, Dordrecht, Netherlands 2000.

[12] J.A. Gamez, S. Moral, and A. Salmeron. *Advances in Bayesian Networks.* Springer, Berlin, Germany 2004

[13] J. Gebhardt. *Learning from Data: Possibilistic Graphical Models.* Habilitation Thesis, University of Braunschweig, Germany 1997

[14] D. Heckerman, D. Geiger, and D.M. Chickering. Learning Bayesian Networks: The Combination of Knowledge and Statistical Data. *Machine Learning* 20:197–243. Kluwer, Dordrecht, Netherlands 1995

[15] F.V. Jensen. *Bayesian Networks and Decision Graphs.* Springer, Berlin, Germany 2001

[16] M.I. Jordan, ed. *Learning in Graphical Models.* MIT Press, Cambridge, MA, USA 1998

[17] S.L. Lauritzen. *Graphical Models.* Oxford University Press, Oxford, United Kingdom 1996

[18] R.E. Neapolitan. *Learning Bayesian Networks.* Prentice Hall, Englewood Cliffs, NJ, USA 2003

[19] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, San Mateo, CA, USA 1988 (2nd edition 1992)

[20] L.K. Rasmussen. *Blood Group Determination of Danish Jersey Cattle in the F-blood Group System (Dina Research Report 8).* Dina Foulum, Tjele, Denmark 1992

[21] M. Singh and M. Valtorta. An Algorithm for the Construction of Bayesian Network Structures from Data. *Proc. 9th Conf. on Uncertainty in Artificial Intelligence (UAI'93, Washington, DC, USA)*, 259–265. Morgan Kaufmann, San Mateo, CA, USA 1993

[22] P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search (Lecture Notes in Statistics 81).* Springer, New York, NY, USA 1993

[23] H. Steck. *Constraint-Based Structural Learning in Bayesian Networks using Finite Data Sets.* Ph.D. thesis, TU München, Germany 2001

[24] J. Whittaker. *Graphical Models in Applied Multivariate Statistics.* J. Wiley & Sons, Chichester, United Kingdom 1990