Identification and Overidentification of Linear Structural Equation Models

Bryant Chen
University of California, Los Angeles
Computer Science Department
Los Angeles, CA, 90095-1596, USA

Algorithm 1 HT-ID(G, Σ, IDEdges)

Initialize: EdgeSets ← all connected edge sets in G
repeat
for each ES in EdgeSets do
    v ← He(ES)
    for each E ⊂ ES such that E ⊄ IDEdges do
        AE ← Allowed(E, IDEdges, G)
        YE ← MaxFlow(G, E, AE)
        if |YE| = |Ta(E)| then
            Identify E using Theorem 1
            IDEdges ← IDEdges ∪ E
        end if
    end for
end for
until All coefficients have been identified or no coefficients have been identified in the last iteration
return IDEdges

Theorem 1. If a g-HT-admissible set for directed edges E_v with head v exists then E_v is identifiable. Further, let Y_E_v = {y_1, ..., y_k} be a g-HT-admissible set for E_v, Ta(E_v) = {p_1, ..., p_k}, and Σ be the covariance matrix of the model variables. Define A as

\[
A_{ij} = \begin{cases} (I - \Lambda)^T \Sigma y_i, & y_i \in htr(v) \text{ or } y_i \text{ connected to } Pa(v) \setminus Ta(E_v), \\ \Sigma y_i, & y_i \notin htr(v) \end{cases} \quad (1)
\]

and b as

\[
b_i = \begin{cases} (I - \Lambda)^T \Sigma y_i, & y_i \in htr(v) \text{ or } y_i \text{ connected to } Pa(v) \setminus Ta(E_v), \\ \Sigma y_i, & y_i \notin htr(v) \end{cases} \quad (2)
\]

Then A is an invertible matrix and A \cdot \Lambda_{Ta(E_v), v} = b.

Proof. The proof for this theorem is similar to the proof of Theorem 1 in Foygel et al. (2012). Rather than giving a complete proof, we simply explain why our changes are valid. The g-HTC identifies arbitrary sets of directed edges belonging to a node rather than all of the directed edges belonging to a node. It is able to do this because of two changes. First, sets that contain nodes that are connected to Pa(v) \setminus Ta(E) via half-treks cannot be half-trek admissible for E (see Definition 6). As a result, the paths from half-trek admissible set, Y_E, to v travel only through coefficients of E and no other coefficients of E. This ensures that A \cdot \Lambda_{Ta(E_v), v} = b. Second, nodes that are connected to Pa(v) \setminus Ta(E) are not allowed unless their coefficients that lie on paths to Pa(v) \setminus Ta(E) are
Algorithm 2 HT-Constraints(\(G, \Sigma, \text{IDEdges}\))

Initialize: \(\text{EdgeSets} \leftarrow \) all connected edge sets in \(G\)
repeat
  for each \(ES\) in \(\text{EdgeSets}\) do
    for each \(E \subset ES\) such that \(E \not\subseteq \text{IDEdges}\) do
      \(A_E \leftarrow \text{Allowed}(E, \text{IDEdges}, G)\)
      \(Y_E \leftarrow \text{MaxFlow}(G, E, A_E)\)
      if \(|Y_E| = |Ta(E)|\) then
        Identify \(E\) using Theorem 1
        \(\text{IDEdges} \leftarrow \text{IDEdges} \cup E\)
      end if
    end for
    for each \(w\) in \(A_E \setminus Y_E\) do
      if \(v \in htr(w)\) then
        Output constraint: \(b_w = a_w \cdot A^{-1}_{\text{right}} \cdot b\)
      else if \(w \notin htr(v)\) then
        Output constraint: \(\Sigma_{wv} = 0\)
      end if
    end for
  end for
until one iteration after all edges are identified or no new edges have been identified in the last iteration
return \(\text{IDEdges}\)
Algorithm 3 decomposes the graph according to its c-components and then applies Algorithm 2 to each sub-model. If there are still unidentified coefficients, then it removes descendant sets and decomposes again. The whole process is repeated until one iteration after every coefficient is identified or no new coefficients are identified in an iteration. $\Sigma_{P_{S_i}}$ is the covariance matrix of $P_{S_i}$, where $S_i$ is a c-component. $\Sigma_{V \setminus D_i}$ is the covariance matrix after marginalizing $D_i$ from $\Sigma$. Finally, $G_{V \setminus D_i}$ is the graph with the set $D_i$ removed.

Algorithm 3 Decompt-HT($G, \Sigma$)

Initialize: IDEdges $\leftarrow \emptyset$
repeat
   IDEdges $\leftarrow$ IDEdges$\cup$Rec-Decomp($G, \Sigma, IDEdges$)
until One iteration after all coefficients have been identified or no coefficients have been identified
return IDEdges

Algorithm 4 Rec-Decomp($G, \Sigma, IDEdges$)

$V \leftarrow$ vertices in $G$
Edges $\leftarrow$ all edges in $G$
for each c-component, $S_i$, in $G$ do
   IDEdges = IDEdges $\cup$ HT-Constraints($G_{S_i}, \Sigma_{S_i}, IDEdges$)
end for
if IDEdges = Edges then
   Return IDEdges
else
   for each descendant set, $D_i$, in $G$ do
      IDEdges $\leftarrow$ IDEdges$\cup$Rec-Decomp($G_{V \setminus D_i}, \Sigma_{V \setminus D_i}, IDEdges$)
   end for
end if
return IDEdges

Theorem 3. Let $M$ be a linear SEM with variables $V$. Let $M'$ be a non-parametric SEM with identical structure to $M$. If the direct effect of $x$ on $z$ for $x, y \in V$ is identified in $M'$ then the coefficient $\lambda_{xy}$ in $M$ is g-HTC identifiable and can be identified using Algorithm 3.

Proof. Let $G$ be the causal graph of $M$ and $M'$. Suppose the direct effect of $x$ on $y$ is identified in $M'$. Then according to Theorem 3 of (Shpitser, 2008), there does not exist a subgraph of $G$ that is a $y$-rooted c-tree (Shpitser, 2008). This implies that $MACS(y) = y$. By recursively decomposing the graph into c-components and marginalizing descendant sets, we can obtain a graph where only $MACS(y)$ and its parents remain in the graph. Since $MACS(Y) = y$, the parents of $y$ in this graph represent a g-HT admissible set that allows the identification of all coefficients of $y$. $\square$

Theorem 4. Any Q-constraint, $Q_S \perp Z$, in a linear SEM, has an equivalent set of HT-constraints that can be discovered using Algorithm 3.

Proof. Consider a Q-constraint, $Q_S$ is not a function of $Z$. This constraint is obtained through some sequence of c-component decomposition and marginalization of descendant sets. In the last step, $Q_S$ is identified from $Q_{SUW}$ for some $W$ such that $Z \subset W \cup Pa(W)$. Let $G' = G_{SUW}$. Now $Q_S$ is not a function of $Z$ implies that $Z \perp_{G'} S | Pa(S)$ since $Z$ must be ordered before $S$ and, therefore, $Z \notin De(S)$. Similarly, $Z \perp_{G'} S | Pa(S)$ implies that $Q_S$ is not a function of $Z$. As a result, the Q-constraint is obtained if and only if $Z \perp_{G'} S | Pa(S)$, where $Z$ is ordered before $S$, and a Q-constraint is equivalent to a conditional independence constraint in the distribution, $P_{SUW}$.

Since pairwise independence implies independence in normal distributions, the constraint $Z \perp S | Pa(S)$ is equivalent to the set of conditional independences, $\{z_i \perp S | Pa(S)\}$, where $z_i \in Z$. We now show that there exists an equivalent HT-constraint for each conditional independence, $z_i \perp S | Pa(S)$ in the distribution $P_{SUW}$. $G'$ is obtainable from recursive c-component decomposition,
and, in \( G' \), \( Pa(S) \) satisfies conditions (i)-(iii) of the g-HTC for the edges from \( Pa(S) \) to \( S \). Additionally, \( z_i \) is not half-trek reachable from \( S \) and either has a half-trek to \( S \) or is separated entirely from \( S \). In both cases, we obtain a HT-constraint that is equivalent to the conditional independence constraint \( Z \perp S | Pa(S) \) in the distribution, \( P_{SUW} \).

If \( z_i \) is separated entirely from \( S \) then the constraint is that \( z_i \) is independent of \( s \). In Algorithm 2, this is exactly the constraint that is outputted. If \( z_i \) is separated from \( S \) by \( Pa(S) \), then Algorithm 2 outputs a constraint that is equivalent to \( z_i \) is independent of \( S \) given \( Pa(S) \). One way to see this is that the conditional covariance matrix of \( \{ z \} \cup S \) given \( Pa(S) \) in \( P_{SUW} \) is the Schur complement of \( \Sigma_{\{ z \} \cup S} \) in \( \Sigma \), where \( \Sigma \) is the covariance matrix of \( \{ z \} \cup S \cup Pa(S) \) in \( P_{SUW} \) and \( \Sigma_{\{ z \} \cup S} \) is the entries of \( \Sigma \) for \( \{ z \} \cup S \). If we rearrange the constraint outputted by Algorithm 2 to read \( b_w - a_w * A_{\text{right}}^{-1} * b = 0 \), then we see that it is simply stating the conditional independence constraint.

\[\text{Lemma 3. Any dormant independence, } x \perp y | w, \text{ do}(Z), \text{ with } x \text{ and } y \text{ singletons has an equivalent Q-constraint.}\]

\[\text{Proof. Let } MACS(Z) \text{ denote the maximal ancestral confounded set of } Z \text{ (Shpitser and Pearl, 2008), the maximal set in which } MACS(Z) = \text{Anc}(Z)_{G(MACS(Z))} = C(Z)_{G(MACS(Z))}, \text{ where } G(MACS(Z)) \text{ is the subgraph of } G \text{ containing only the variables in } MACS(Z) \text{ and } C(Z) \text{ is the c-component of } Z.\]

According to Theorem 6 of Shpitser and Pearl (2008), there exists a dormant independence between singletons, \( x \) and \( y \), if and only if \( x \) is not a parent of \( MACS(y) \), \( y \) is not a parent of \( MACS(x) \), and there is no bidirected arc between \( MACS(x) \) and \( MACS(y) \). In this case, \( x \perp y | \text{do}(MACS(x) \cup MACS(y)), (MACS(x) \cup MACS(y)) \setminus \{x, y\} \). Now, it is not hard to show using results from (Tian, 2002) that \( Q_{MACS(y)} \) is identifiable. Further, since there is no bidirected arc between \( MACS(x) \) and \( MACS(y) \), it is possible to identify \( Q_{MACS(y)} \) without marginalizing over \( x \). Finally, we know that \( x \notin Pa(MACS(y)) \) so we obtain the Q-constraint, \( Q_{MACS(y)} \) is not a function of \( x \).

Now, we will show that the Q-constraint, \( Q_{MACS(y)} \perp x \) also implies the dormant independence, \( x \perp y | \text{do}(MACS(x) \cup MACS(y)), (MACS(x) \cup MACS(y)) \setminus \{x, y\} \). In proving Theorem 5, we showed that \( Q_{MACS(y)} \perp x \) implies that \( y \perp x | Pa(S) \) in some distribution, \( Q_{SUW} \), where \( y \in S \) and \( x \in W \). Recalling that a Q-factor is just an interventional distribution, we have a dormant independence between \( x \) and \( y \). Since Theorem 6 of (Shpitser and Pearl, 2008) gives a necessary and sufficient condition for dormant independence between \( x \) and \( y \), we also have that \( x \perp y | \text{do}(MACS(x) \cup MACS(y)), (MACS(x) \cup MACS(y)) \setminus \{x, y\} \). \( \square \)

\[\text{References}\]


