Exact Constraint-based Causal Discovery

Antti Hyttinen

Joint work with Matti Järvisalo, Paul Saikko

University of Helsinki, Department of Computer Science,
Helsinki Institute for Information Technology

DALI 2017, Tenerife
20.4.2017
1 Introduction

2 Implicit Hitting Set Approach

3 Dseptor: Utilizing 3 Domain Specific Techniques

4 Experiments

5 Conclusion
**Input:** Data.

**Task:** Find an equivalence class of causal graph structures that may have generated the data.
**Input:** Data.

**Task:** Find an equivalence class of causal graph structures that may have generated the data.

**Score-based Methods:** [Cooper, Heckerman,...]
- Maximize Bayesian marginal likelihood or BIC.
- Good Accuracy, limited model space.
- **Exact:** Find the globally optimal graph(s).
**Input:** Data.

**Task:** Find an equivalence class of causal graph structures that may have generated the data.

**Score-based Methods:** [Cooper, Heckerman,...]
- Maximize Bayesian marginal likelihood or BIC.
- Good Accuracy, limited model space.
- **Exact:** Find the globally optimal graph(s).

**Constraint-based Methods:** [Pearl, Spirtes,...]
- Deduce the graph structure from independence test results.
- General of the model space: Latent confounders, Cycles.
- Scale up by making **greedy** decision $\Rightarrow$ poor accuracy.
**INPUT:** All weighted cond. (in)dependencies $K$ among vars.

**TASK:** Find $G$ that minimizes $\sum_{k \in K : G \not\models k} w(k)$. 

---

**Related Work:**

- Hyttinen et al. ’14
- Magliacane et al. ’16
- Borboudakis et al. ’16

**Several options for getting weighted independence constraints.**

- Margaritis et al. ’09
- Claassen et al. ’12
- Triantafillou et al. ’15
- Magliacane et al. ’16
**Input:** All weighted cond. (in)dependencies $K$ among vars.

**Task:** Find $G$ that minimizes $\sum_{k \in K : G \nmid k} w(k)$.

- Through assuming Causal Markov and Faithfulness:
  $$X \perp \perp Y \mid C \iff X \text{ is d-separated from } Y \text{ given } C$$

[Hyttinen et al. '14, Magliacane et al. '16, Borboudakis et al. '16]
**Input:** All weighted cond. (in)dependencies $K$ among vars.

**Task:** Find $G$ that minimizes $\sum_{k \in K : G \nvDash k} w(k)$.

- Through assuming Causal Markov and Faithfulness:
  
  $X \perp\!\!\!\!\perp Y \mid C \iff X \text{ is d-separated from } Y \text{ given } C$

- A very hard optimization problem, but can be solved **exactly** using off-the-shelf Boolean opt. solvers (e.g. wMaxSAT).
**Exact Constraint-based Causal Discovery** [Hyttinen et al. ’14]

**Input:** All weighted cond. (in)dependencies $K$ among vars.

**Task:** Find $G$ that minimizes $\sum_{k \in K : G \not\models k} w(k)$.

- Through assuming Causal Markov and Faithfulness:
  
  $X \perp \perp Y \mid C \iff X$ is d-separated from $Y$ given $C$

- A very hard optimization problem, but can be solved **exactly**
  using off-the-shelf Boolean opt. solvers (e.g. wMaxSAT).

- Often accurate. [Hyttinen et al. ’14, Magliacane et al. ’16, Borboudakis et al. ’16]

- Several options for getting weighted independence constraints.
**Exact Constraint-based Causal Discovery** [Hyttinen et al. '14]

**Input:** All weighted cond. (in)dependencies $K$ among vars.

**Task:** Find $G$ that minimizes $\sum_{k \in K} : G \not\perp \parallel k \ w(k)$.

- Through assuming Causal Markov and Faithfulness:
  \[ X \perp \perp Y \mid C \iff X \text{ is d-separated from } Y \text{ given } C \]
- A very hard optimization problem, but can be solved **exactly** using off-the-shelf Boolean opt. solvers (e.g. wMaxSAT).
- Often accurate. [Hyttinen et al. '14, Magliacane et al. '16, Borboudakis et al. '16]
- Several options for getting weighted independence constraints.
  [Margaritis et al. '09, Claassen et al. '12, Triantafillou et al. '15, Magliacane et al. '16]
- Related Work: [Magliacane et al. '16, Claassen et al. '12, Triantafillou et al. '15]
Implicit Hitting Set Approach
Implicit Hitting Set Approach [Davies '13]

SAT-solver:
• Is there a graph that satisfies constraints $K$?
• CNF-encoding of d-separation/connection: $X \not\perp \perp Y \mid C \iff X \rightarrow Y \lor Y \rightarrow X \lor X \leftrightarrow Y \lor \ldots$
• Returns SAT and a solution, or UNSAT and a core.
• Core is a subset of constraints not simultaneously satisfiable.

Integer Programming -solver:
• Finds a hitting set $H$ that minimizes $\sum_{k \in H} w(k)$, s.t. $H \cap c \neq \emptyset$ for all cores $c$.

Implementation:
• LMHS by [Saikko et al. '16] uses MiniSAT (backtracking depth first search) and CPLEX (simplex-based branch-and-cut).
Implicit Hitting Set Approach [Davies ’13]

**SAT-solver:**
- Is there a graph that satisfies constraints $K$?
- CNF-encoding of d-separation/connection: [Hyttinen et al. ’14]

\[
X \not\perp \perp Y \mid C \iff X \rightarrow Y \lor Y \rightarrow X \lor X \leftrightarrow Y \lor \ldots
\]

- Returns SAT and a solution, or UNSAT and a **core**.
- **Core** is a subset of constraints not simultaneously satisfiable.
Implicit Hitting Set Approach [Davies '13]

SAT-solver:
- Is there a graph that satisfies constraints $K$?
- CNF-encoding of d-separation/connection: [Hyttinen et al. '14]

\[ X \not\perp \perp Y | C \iff X \rightarrow Y \lor Y \rightarrow X \lor X \leftrightarrow Y \lor \ldots \]

- Returns SAT and a solution, or UNSAT and a core.
- Core is a subset of constraints not simultaneously satisfiable.

Integer Programming -solver:
- Finds a hitting set $H$ that minimizes $\sum_{k \in H} w(k)$, s.t. $H \cap c \neq \emptyset$ for all cores $c$. 

Implementations:
- LMHS by [Saikko et al. '16] uses MiniSAT (backtracking depth first search) and CPLEX (simplex-based branch-and-cut).
Implicit Hitting Set Approach  [Davies '13]

**SAT-solver:**
- Is there a graph that satisfies constraints $K$?
- CNF-encoding of d-separation/connection: [Hyttinen et al. '14]
  
  \[ X \not\perp \perp Y | C \iff X \rightarrow Y \lor Y \rightarrow X \lor X \leftrightarrow Y \lor \ldots \]
- Returns SAT and a solution, or UNSAT and a **core**.
- **Core** is a subset of constraints not simultaneously satisfiable.

**Integer Programming -solver:**
- Finds a hitting set $H$ that minimizes \( \sum_{k \in H} w(k) \),
  
  \[ \text{s.t. } H \cap c \neq \emptyset \text{ for all cores } c. \]

**Implementation:**
- LMHS by [Saikko et al. '16] uses MiniSAT (backtracking depth first search) and CPLEX (simplex-based branch-and-cut).
Implicit Hitting Set Approach on a Toy Example

<table>
<thead>
<tr>
<th>Constraints</th>
<th>Weights</th>
</tr>
</thead>
<tbody>
<tr>
<td>( Q \not\perp X )</td>
<td>SAT solver</td>
</tr>
<tr>
<td>( Y \not\perp Z</td>
<td>X, W )</td>
</tr>
<tr>
<td>( X \not\perp Z</td>
<td>Y, W )</td>
</tr>
<tr>
<td>( Q \perp Z )</td>
<td>IP solver</td>
</tr>
<tr>
<td>( Y \not\perp Z</td>
<td>W )</td>
</tr>
<tr>
<td>( X \perp Y</td>
<td>Z, W )</td>
</tr>
<tr>
<td>( X \perp Y</td>
<td>W )</td>
</tr>
<tr>
<td>( X \not\perp Z</td>
<td>W )</td>
</tr>
<tr>
<td>( X \not\perp W )</td>
<td></td>
</tr>
<tr>
<td>( X \not\perp W</td>
<td>Z )</td>
</tr>
<tr>
<td>( X \not\perp Y</td>
<td>W, Q )</td>
</tr>
<tr>
<td>( Q \not\perp Y</td>
<td>X )</td>
</tr>
<tr>
<td>( W \not\perp Z )</td>
<td></td>
</tr>
<tr>
<td>( Q \perp W )</td>
<td></td>
</tr>
<tr>
<td>( Y \perp Q</td>
<td>W )</td>
</tr>
<tr>
<td>( Q \perp Z</td>
<td>W )</td>
</tr>
</tbody>
</table>

Step 1 \( L = 0, U = 16 \)
Implicit Hitting Set Approach on a Toy Example

\[
\begin{align*}
Q \not\perp X & \quad Y \not\perp Z | X, W & \quad X \not\perp Z | Y, W & \quad Q \perp Z \\
Y \not\perp Z | W & \quad X \perp Y | Z, W & \quad X \perp Y | W & \quad X \not\perp Z | W \\
X \not\perp W & \quad X \not\perp W | Z & \quad X \not\perp Y | W, Q & \quad Q \not\perp Y | X \\
W \not\perp Z & \quad Q \perp W & \quad Y \perp Q | W & \quad Q \perp Z | W
\end{align*}
\]

SAT solver

IP solver

Step 2 $L = 0, U = 16$
Implicit Hitting Set Approach on a Toy Example

\[
\begin{align*}
Q \not\perp X & \quad Y \not\perp Z|X, W & \quad X \not\perp Z|Y, W & \quad Q \perp Z \\
Y \not\perp Z|W & \quad X \perp Y|Z, W & \quad X \perp Y|W & \quad X \not\perp Z|W \\
X \not\perp W & \quad X \not\perp W|Z & \quad X \not\perp Y|W, Q & \quad Q \not\perp Y|X \\
W \not\perp Z & \quad Q \perp W & \quad Y \perp Q|W & \quad Q \perp Z|W
\end{align*}
\]

SAT solver \quad \text{a core} \quad \text{IP solver}

Step 3 \quad L = 0, U = 16
Implicit Hitting Set Approach on a Toy Example

\[
\begin{align*}
Q \not\parallel X & \quad Y \not\parallel Z|X, W \quad X \not\parallel Z|Y, W \quad Q \not\parallel Z \\
Y \not\parallel Z|W & \quad X \not\parallel Y|Z, W \quad X \not\parallel Y|W \quad X \not\parallel Z|W \\
X \not\parallel W & \quad X \not\parallel W|Z \quad X \not\parallel Y|W, Q \quad Q \not\parallel Y|X \\
W \not\parallel Z & \quad Q \not\parallel W \quad Y \not\parallel Q|W \quad Q \not\parallel Z|W
\end{align*}
\]

- SAT solver
- IP solver

Step 4 \( L = 0, U = 16 \)
Implicit Hitting Set Approach on a Toy Example

Step 5 $L = 1, U = 16$
Implicit Hitting Set Approach on a Toy Example

\[
\begin{align*}
Q \not\perp X & \quad Y \not\perp Z \mid X, W & \quad X \not\perp Z \mid Y, W & \quad Q \perp Z \\
X \perp Y \mid Z, W & \quad X \perp Y \mid W & \quad X \not\perp Z \mid W \\
X \not\perp W & \quad X \not\perp W \mid Z & \quad X \not\perp Y \mid W, Q & \quad Q \not\perp Y \mid X \\
W \not\perp Z & \quad Q \perp W & \quad Y \perp Q \mid W & \quad Q \perp Z \mid W
\end{align*}
\]

SAT solver

IP solver

Step 6 \( L = 1, U = 16 \)
Implicit Hitting Set Approach on a Toy Example

\[
Q \not\perp X \\
Y \not\perp Z | X, W \\
X \not\perp Z | Y, W \\
Q \perp Z \\
X \perp Y | Z, W \\
X \perp Y | W \\
X \not\perp Z | W \\
X \not\perp W \\
X \not\perp W | Z \\
X \not\perp Y | W, Q \\
Q \not\perp Y | X \\
W \not\perp Z \\
Q \perp W \\
Y \perp Q | W \\
Q \perp Z | W
\]

SAT solver \quad IP solver

a core

UNSAT

Step \quad 7 \quad L = 1, \ U = 16
Implicit Hitting Set Approach on a Toy Example

\[
\begin{align*}
Q \not\perp X & \quad Y \not\perp Z|X, W & \quad X \not\perp Z|Y, W & \quad Q \perp Z \\
Y \not\perp Z|W & \quad X \perp Y|Z, W & \quad X \perp Y|W & \quad X \not\perp Z|W \\
X \not\perp W & \quad X \not\perp W|Z & \quad X \not\perp Y|W, Q & \quad Q \not\perp Y|X \\
W \not\perp Z & \quad Q \perp W & \quad Y \perp Q|W & \quad Q \perp Z|W \\
\end{align*}
\]

SAT solver

IP solver

Step 8  \( L = 1, U = 16 \)
Implicit Hitting Set Approach on a Toy Example

Step 9 \( L = 1, U = 16 \)
Implicit Hitting Set Approach on a Toy Example

\[
\begin{align*}
Q \not\perp X & \quad Y \not\perp Z|X, W & \quad X \not\perp Z|Y, W & \quad Q \perp Z \\
Y \not\perp Z|W & \quad X \perp Y|Z, W & \quad X \not\perp Z|W \\
X \not\perp W & \quad X \not\perp W|Z & \quad X \not\perp Y|W, Q & \quad Q \not\perp Y|X \\
W \not\perp Z & \quad Q \perp W & \quad Y \perp Q|W & \quad Q \perp Z|W
\end{align*}
\]

SAT solver

IP solver

Step 10 $L = 1$, $U = 16$
### Implicit Hitting Set Approach on a Toy Example

| \( Q \not\perp X \) | \( Y \not\perp Z|X, W \) | \( X \not\perp Z|Y, W \) | \( Q \perp Z \) |
|---------------------|-----------------------------|-----------------------------|---------------------|
| \( Y \not\perp Z|W \) | \( X \perp Y|Z, W \) | \( X \not\perp Z|W \) | \( X \not\perp Z|W \) |
| \( X \not\perp W \) | \( X \not\perp W|Z \) | \( X \not\perp Y|W, Q \) | \( Q \not\perp Y|X \) |
| \( W \not\perp Z \) | \( Q \perp W \) | \( Y \perp Q|W \) | \( Q \perp Z|W \) |

#### Step 11

\( L = 1, U = 1 \)
Implicit Hitting Set Approach on a Toy Example

Step 12 \( L = 1, U = 1 \)
Implicit Hitting Set Approach on a Toy Example

Step 13 $L = 1, U = 1$
Dseptor: Utilizing 3 Domain Specific Techniques
Each extracted core call requires potentially many (NP-)SAT-solver calls. How to avoid some of these?

• Instead, we can find general patterns of cores:
  • Observe which cores the solver uses, e.g. 
    \{X \perp \perp Z; X \not\perp \perp Z | Y; X \perp \perp Y | C\}.
  • Generalize a core to a pattern, e.g.
    \{X \perp \perp Z | C; X \not\perp \perp Z | Y, C; X \perp \perp Y | C\}, \forall X, Y, Z, C.
  • Prove that the pattern generally gives a (minimal) core.

• We identified 7 cores patterns among 3-4 nodes and 3-5 constraints, each including independencies and dependencies.

Benefit: Thousands of small cores can be found in a fraction of the total solving time.
Each extracted core call requires potentially many (NP-)SAT-solver calls. How to avoid some of these?

- Instead, we can find general patterns of cores:
  - **Observe** which cores the solver uses, e.g.
    \[ \{X \perp Z; \; X \not\perp Z | Y; \; X \perp Y\} \]
  
  - **Generalize** a core to a pattern, e.g.
    \[ \{X \perp Z | C; \; X \not\perp Z | Y, C; \; X \perp Y | C\}, \; \forall X, Y, Z, C \]

- **Prove** that the pattern generally gives a (minimal) core.
Each extracted core call requires potentially many (NP-)SAT-solver calls. How to avoid some of these?

- Instead, we can find general patterns of cores:
  - **Observe** which cores the solver uses, e.g.

\[
\{ X \perp Z; \quad X \not\perp Z|Y; \quad X \perp Y \}\]

- **Generalize** a core to a pattern, e.g.

\[
\{ X \perp Z|C; \quad X \not\perp Z|Y, C; \quad X \perp Y|C\}, \quad \forall X, Y, Z, C
\]

- **Prove** that the pattern generally gives a (minimal) core.

- We identified 7 cores patterns among 3-4 nodes and 3-5 constraints, each including independencies and dependencies.
Each extracted core call requires potentially many (NP-)SAT-solver calls. How to avoid some of these?

- Instead, we can find general patterns of cores:
  - **Observe** which cores the solver uses, e.g.
    \[
    \{X \perp Z; \ X \not\perp Z|Y; \ X \perp Y\}
    \]
  - **Generalize** a core to a pattern, e.g.
    \[
    \{X \perp Z|C; \ X \not\perp Z|Y, C; \ X \perp Y|C\}, \ \forall X, Y, Z, C
    \]
  - **Prove** that the pattern generally gives a (minimal) core.

- We identified 7 cores patterns among 3-4 nodes and 3-5 constraints, each including independencies and dependencies.

**Benefit:** Thousands of small cores can be found in a fraction of the total solving time.
For all instantiations of nodes $X, Y, Z, W$ and set $C$:

(i) $\{X \perp Z|C; \ X \perp Y|C; \ X \not\perp Z|Y, C\}$

(ii) $\{X \not\perp Z|C; \ Y \not\perp Z|C; \ X \perp Y|C; \ X \perp Y|Z, C\}$

(iii) $\{X \not\perp Z|Y, C; \ Y \not\perp Z|X, C; \ X \perp Y|C; \ X \perp Y|Z, C\}$

(iv) $\{Y \not\perp Z|C; \ X \not\perp Z|C; \ Z \perp W|X, Y, C; \ X \perp Y|Z, C; \ X \perp Y|W, C\}$

(v) $\{Y \not\perp Z|C; \ X \not\perp Z|C; \ Z \perp W|Y, C; \ X \perp Y|Z, C; \ X \perp Y|W, C\}$

(vi) $\{X \not\perp Y|Z, C; \ Y \not\perp Z|X, W, C; \ W \not\perp Y|Z, C; \ W \perp X|Y, Z, C; \ X \perp Z|W, C\}$

(vii) $\{X \not\perp Y|Z, C; \ Y \not\perp Z|X, W, C; \ W \not\perp Y|C; \ W \perp X|Y, C; \ X \perp Z|W, C\}$

are minimal cores.
Technique 2: Incremental Core Extraction

Plain IHS-approach tends to produce large cores. How find diverse and disjoint cores for exact causal discovery?

• Plain IHS-approach enforces all constraints, except $H$.
• Instead, we can input constraints one by one, and check satisfiability. [See Triantafillou et al. '15]
• Which order? random order (currently).
  Benefits: Smaller diverse cores, good upper bounds.
Plain IHS-approach tends to produce large cores. How find diverse and disjoint cores for exact causal discovery?
Plain IHS-approach tends to produce large cores. How find diverse and disjoint cores for exact causal discovery?

• Plain IHS-approach enforces all constraints, except $H$. 

\[ \begin{align*}
Q \perp X & \quad \quad Y \perp Z|X, W \quad X \perp Z|Y, W \quad Q \perp Z \\
Y \perp Z|W & \quad \quad X \perp Y|Z, W \quad X \perp Y|W \quad X \perp Z|W \\
X \perp W & \quad \quad X \perp W|Z \quad X \perp Y|W, Q \quad Q \perp Y|X \\
W \perp Z & \quad \quad Q \perp W \quad Y \perp Q|W \quad Q \perp Z|W
\end{align*} \]
Plain IHS-approach tends to produce large cores. How find diverse and disjoint cores for exact causal discovery?

- Plain IHS-approach enforces all constraints, except $H$.
- Instead, we can input constraints one by one, and check satisfiability. [See Triantafillou et al. '15]
- Which order?
Technique 2: Incremental Core Extraction

Plain IHS-approach tends to produce large cores. How find diverse and disjoint cores for exact causal discovery?

- Plain IHS-approach enforces all constraints, except $H$.
- Instead, we can input constraints one by one, and check satisfiability. [See Triantafillou et al. ’15]
- Which order? — random order (currently).
Plain IHS-approach tends to produce large cores. How find diverse and disjoint cores for exact causal discovery?

- Plain IHS-approach enforces all constraints, except $H$.
- Instead, we can input constraints one by one, and check satisfiability. [See Triantafillou et al. ’15]
- Which order? — random order (currently).

**Benefits:** Smaller diverse cores, good upper bounds.
How can we exploit sparseness without losing exactness?
How can we exploit sparseness without losing exactness?

\[
\begin{align*}
Q \not\perp X & \quad Y \not\perp Z | X, W & \quad X \not\perp Z | Y, W & \quad Q \perp Z \\
Y \not\perp Z | W & \quad X \perp Y | Z, W & \quad X \perp Y | W & \quad X \not\perp Z | W \\
X \not\perp W & \quad X \not\perp W | Z & \quad X \not\perp Y | W, Q & \quad Q \not\perp Y | X \\
W \not\perp Z & \quad Q \perp W & \quad Y \perp Q | W & \quad Q \perp Z | W
\end{align*}
\]
Technique 3: Bounds-based Constraint Hardening

How can we exploit sparseness without losing exactness?

- A graph with $Q - Z$ violates $\forall C : Q \perp Z|C$ & hits the cores.
How can we exploit sparseness without losing exactness?

- A graph with $Q \perp Z$ violates $\forall C : Q \perp Z | C$ & hits the cores.
- Thus, $L' = 3$ is a lower bound for all such graphs. If $U < L'$, the absence can be hardened. LP-bound is enough!
Technique 3: Bounds-based Constraint Hardening

How can we exploit sparseness without losing exactness?

- A graph with $Q \perp Z$ violates $\forall C : Q \perp Z|C$ & hits the cores.
- Thus, $L' = 3$ is a lower bound for all such graphs. If $U < L'$, the absence can be hardened. LP-bound is enough!
- In contrast: PC makes the hard decision of non-adjacency of $Q, Z$ from a single independence $Q \perp Z|S$. 
Technique 3: Bounds-based Constraint Hardening

How can we exploit sparseness without losing exactness?

- A graph with $Q \perp Z$ violates $\forall C : Q \perp Z|C$ & hits the cores.
- Thus, $L' = 3$ is a lower bound for all such graphs. If $U < L'$, the absence can be hardened. LP-bound is enough!
- In contrast: PC makes the hard decision of non-adjacency of $Q, Z$ from a single independence $Q \perp Z|S$.

Benefits: Less soft constraints, SAT-instances are tighter.
Experiments
Example Run of Dseptor

Real data set #22 (7 vars of StatLog)

- Dseptor UB
- Dseptor LB
- LB by domain specific cores
- Dseptor hardened constraints
- Dseptor hardened edge abs.
- Plain LMHS LB
Running Time Performance (1)

Synthetic data, 100 instances, 7 nodes, 672 soft constraints

- Dseptor
  - w.o. hard constraints
  - w.o. incr. core extr.
  - w.o. d.s. cores
- Maxino
- LMHS
- Clingo

0 20 40 60 80 100
0 100 200 300 400 500 600
instances (sorted for each line)
solving time per instance (s)
Running Time Performance (1)

Synthetic data, 100 instances, 7 nodes, 672 soft constraints

- Dseptor
- w.o. hards
- w.o. incr.
- w.o. cores
- Maxino
- LMHS
- QMaxSAT
- Clingo
- CPLEX
- OpenWBO
- MSCG
- MaxHS
- wpm3
- Virtual Best

instances (sorted for each line)
solving time per instance (s)
Running Time Performance (2)

Real-world data, 6–10 nodes, 240–11520 soft constraints

- Dseptor
- Maxino
- LMHS

instances (sorted for each line)
solving time per instance (s)
Conclusion
Conclusion

The presentation included:

• Exact Constraint-based Causal Discovery
• A new approach for solving the optimization problem
• Exploiting a general MaxSAT solver and
  1. Domain specific cores
  2. Incremental core extraction
  3. Bounds-based constraint hardening
• Faster running time performance

Open Questions:
• Are further running time improvements possible?
• Building in more general constraints?
• What to compromise without losing power & accuracy?

Thanks!
Conclusion

The presentation included:

- Exact Constraint-based Causal Discovery
- A new approach for solving the optimization problem
- Exploiting a general MaxSAT solver and
  1. Domain specific cores
  2. Incremental core extraction
  3. Bounds-based constraint hardening
- Faster running time performance

Open Questions:

- Are further running time improvements possible?
- Building in more general constraints?
- What to compromise without losing power & accuracy?
The presentation included:

- Exact Constraint-based Causal Discovery
- A new approach for solving the optimization problem
- Exploiting a general MaxSAT solver and
  1. Domain specific cores
  2. Incremental core extraction
  3. Bounds-based constraint hardening
- Faster running time performance

Open Questions:

- Are further running time improvements possible?
- Building in more general constraints?
- What to compromise without losing power & accuracy?

Thanks!
Assumptions

Causal Markov

\[ X \perp_{G} Y \mid C \]

\[ X \not\perp_{G} Z \mid C \]

Causal graph

\( G \)

Distribution

\[ X \perp Y \mid C \]

\[ X \not\perp Z \mid C \]

Faithfulness

Finite sample effects

Data

\( X \)