Algorithms for Reasoning with graphical models

Class3
Rina Dechter
Road Map

- Graphical models
- Constraint networks Model
- Inference
  - Variable elimination for Constraints
  - Variable elimination for CNFs
  - Variable elimination for Linear Inequalities
  - Constraint propagation
- Search
- Probabilistic Networks
Inference: Join and Project

- Given 2 constraints we can deduce a new one by join and then project, via variable-elimination

**Join operation** over $A$ finds all solutions satisfying constraints that involve $A$

\[
\begin{align*}
R_{AB} & \quad \begin{array}{cc}
A & B \\
r & g \\
g & r
\end{array} \\
R_{AC} & \quad \begin{array}{cc}
A & C \\
r & g \\
g & r
\end{array}
\end{align*}
\]

Join \( R_{AB}, R_{AC} \)

Project on $BC$

\[
\begin{align*}
R_{BC} & \quad \begin{array}{ccc}
B & A & C \\
r & g & r \\
g & r & g
\end{array}
\end{align*}
\]
Bucket Elimination
Adaptive Consistency (Dechter & Pearl, 1987)

Bucket E: \( E \neq D, \ E \neq C \)
Bucket D: \( D \neq A \)
Bucket C: \( C \neq B \)
Bucket B: \( B \neq A \)
Bucket A: contradiction

Complexity: \( O(n \exp(w^*)) \)
\( w^* \) - induced width
The Idea of Elimination

What is the Inferred function?

\[ R_{DBC} = \prod_{DBC} R_{ED} \bowtie R_{EB} \bowtie R_{EC} \]

Eliminate variable E \(\Leftrightarrow\) join and project
Bucket-Elimination

Bucket(E): E \neq D, E \neq C, E \neq B
Bucket(D): D \neq A \parallel R_{DCB}
Bucket(C): C \neq B \parallel R_{ACB}
Bucket(B): B \neq A \parallel R_{AB}
Bucket(A): \quad R_A

Bucket(A): A \neq D, A \neq B
Bucket(D): D \neq E \parallel R_{DB}
Bucket(C): C \neq B, C \neq E
Bucket(B): B \neq E \parallel R^{D}_{BE}, R^{C}_{BE}
Bucket(E): \quad R_E

Complexity: O(n \exp(w^*(d))),
w^*(d) - induced width along ordering d
The Induced-Width

- **Width along** \(d\), \(w(d)\):
  - max # of previous parents

- **Induced width** \(w^*(d)\):
  - The width in the ordered induced graph

- **Induced-width** \(w^*\):
  - Smallest induced-width over all orderings

- **Finding** \(w^*\)
  - NP-complete \((Arnborg, 1985)\) but greedy heuristics \((\text{min-fill})\).
**Initialize:** partition constraints into $\text{bucket}_1, \ldots, \text{bucket}_n$

**For** $i = n$ down to 1 along d // process in reverse order

**for** all relations $R_1, \ldots, R_m \in \text{bucket}_i$ **do**

join and “project-out” $X_i$

$$R_{\text{new}} \leftarrow \prod_{(-X_i)} (-X_i) \big( \bigwedge_j R_j \big)$$

**If** $R_{\text{new}}$ is not empty, add it to $\text{bucket}_k$, $k < i$, **where** $k$ **is** the largest variable index in $R_{\text{new}}$

**Else** problem is unsatisfiable

**Return** the set of all relations (old and new) in the buckets
Properties of Adaptive-Consistency

- Adaptive consistency generates a constraint network that is **backtrack-free** (can be solved without dead-ends).

**Definition 3.1.2 (partial solution)** Given a constraint network \( \mathcal{R} \), we say that an assignment of values to a subset of the variables \( S = \{X_1, ..., X_j\} \) given by \( \bar{a} = (\langle X_1, a_1 \rangle, \langle X_2, a_2 \rangle, ..., \langle X_j, a_j \rangle) \) is consistent relative to \( \mathcal{R} \) iff it satisfies every constraint whose scope is subsumed in \( S \). The assignment \( \bar{a} \) is also called a **partial solution** of \( \mathcal{R} \).

**Definition 3.1.3 (backtrack-free search)** A constraint network is **backtrack-free** relative to a given ordering \( d = (X_1, ..., X_n) \) if for every \( i \leq n \), every partial solution over \( (X_1, ..., X_i) \) can be consistently extended to include \( X_{i+1} \).
Properties of Adaptive-Consistency (AC)

- Adaptive consistency generates a constraint network that is \textbf{backtrack-free} (can be solved without dead-ends).

- The time and space complexity of AC along ordering $d$ is exponential in $w^*(d)$.

\textbf{Theorem 3.9} \textit{The time and space complexity of ADAPTIVE-CONSISTENCY is $O((r + n)k^{w^*(d)+1})$ and $O(n \cdot k^{w^*(d)})$, respectively, where $n$ is the number of variables, $k$ is the maximum domain size, and $w^*(d)$ is the induced-width along the order of processing $d$ and $r$ is the number of the problems' constraints.}
Example: deadends, backtrack-freeness

Assign values in the order D,B,C,A before and after adaptive-consistence

Order A,B,C,D, order A,B,D,C
Properties of Adaptive-Consistency

- Adaptive-consistency generates a constraint network that is **backtrack-free** (can be solved without dead-ends).

- The time and space complexity of adaptive-consistency along ordering $d$ is time and memory exponential in $w^*(d)$.

- Therefore, problems having **bounded induced-width** are tractable (solved in polynomial time).
  - **trees** ($w^*=1$),
  - **series-parallel networks** ($w^*=2$),
  - and in general **$k$-trees** ($w^*=k$).
Solving Trees

(Mackworth and Freuder, 1985)

Adaptive consistency is linear for trees and equivalent to enforcing \textit{directional arc-consistency} (recording only unary constraints)
Tree Solving is Easy
Tree Solving is Easy
Tree Solving is Easy
Tree Solving is Easy
Tree Solving is Easy

1, 2, 3

1, 2

<

X

1, 2

<

<

1, 2, 3

1, 2, 3

1, 2, 3

1, 2, 3

1, 2, 3

T

Z

R

S

U

class2 828X 2019
Tree Solving is Easy
Tree Solving is Easy
Tree Solving is Easy

Adaptive-consistency is linear time because induced-width is 1
(Constraint propagation Solves trees in linear time)
Example: Crossword Puzzle

\[ R_{1,2,3,4,5} = \{(H, O, S, E, S), (L, A, S, E, R), (S, H, E, E, T), (S, N, A, I, L), (S, T, E, E, R)\} \]
\[ R_{3,6,9,12} = \{(H, I, K, E), (A, R, O, N), (K, E, E, T), (E, A, R, N), (S, A, M, E)\} \]
\[ R_{5,7,11} = \{(R, U, N), (S, U, N), (L, E, T), (Y, E, S), (E, A, T), (T, E, N)\} \]
\[ R_{8,9,10,11} = R_{3,6,9,12} \]
\[ R_{10,13} = \{(N, O), (B, E), (U, S), (I, T)\} \]
\[ R_{12,13} = R_{10,13} \]
Adaptive-Consistency on the Crossword Puzzle
Adaptive-Consistency on the Crossword Puzzle

\[
\begin{align*}
R_{1,2,3,4,5} &= \{(H, O, S, E, S), (L, A, S, E, R), (S, H, E, E, T), \\
&\quad (S, N, A, I, L), (S, T, E, E, R)\} \\
R_{3,6,9,12} &= \{(H, I, K, E), (A, R, O, N), (K, E, E, T), (E, A, R, N), \\
&\quad (S, A, M, E)\} \\
R_{5,7,11} &= \{(R, U, N), (S, U, N), (L, E, T), (Y, E, S), (E, A, T), (T, E, N)\} \\
R_{8,9,10,11} &= R_{3,6,9,12} \\
R_{10,13} &= \{(N, O), (B, E), (U, S), (I, T)\} \\
R_{12,13} &= R_{10,13}
\end{align*}
\]
Adaptive-Consistency on the Crossword Puzzle

\[ R_{1,2,3,4,5} = \{(H, O, S, E, S), (L, A, S, E, R), (S, H, E, E, T), (S, N, A, I, L), (S, T, E, E, R)\} \]
\[ R_{3,6,9,12} = \{(H, I, K, E), (A, R, O, N), (K, E, E, T), (E, A, R, N), (S, A, M, E)\} \]
\[ R_{5,7,11} = \{(R, U, N), (S, U, N), (L, E, T), (Y, E, S), (E, A, T), (T, E, N)\} \]
\[ R_{8,9,10,11} = R_{3,6,9,12} \]
\[ R_{10,13} = \{(N, O), (B, E), (U, S), (I, T)\} \]
\[ R_{12,13} = R_{10,13} \]
Road Map

- Graphical models
- Constraint networks Model
- Inference
  - Variable elimination for Constraints
  - Variable elimination for CNFs
  - Variable elimination for Linear Inequalities
  - Constraint propagation
- Search
- Probabilistic Networks
Gaussian and Boolean Propagation, Resolution

- Linear inequalities
  \[ x + y + z \leq 15, \ z \geq 13 \Rightarrow x \leq 2, \ y \leq 2 \]

- Boolean constraint propagation, unit resolution
  \[(A \lor B \lor \neg C), (\neg B) \Rightarrow (A \lor \neg C)\]
Definition 3.2.1 (extended composition) The extended composition of relation \( R_{S_1}, \ldots, R_{S_m} \) relative to a subset of variables \( A \subseteq \bigcup_{i=1}^{m} S_i \), denoted \( EC_A(R_{S_1}, \ldots, R_{S_m}) \), is defined by

\[
EC_A(R_{S_1}, \ldots, R_{S_m}) = \pi_A(\bigotimes_{i=1}^{m} R_{S_i})
\]

Example 3.2.2 Consider the two clauses \( \alpha = (P \lor \neg Q \lor \neg O) \) and \( \beta = (Q \lor \neg W) \). Now let the relation \( R_{PQO} = \{000, 100, 010, 001, 110, 101, 111\} \) be the models of \( \alpha \) and the relation \( R_{QW} = \{00, 10, 11\} \) be the models of \( \beta \). Resolving these two clauses over \( Q \) generates the resolvent clause \( \gamma = res(\alpha, \beta) = (P \lor \neg O \lor \neg W) \). The models of \( \gamma \) are \( \{(000, 100, 010, 001, 110, 101, 111)\} \). It is easy to see that \( EC_{PQW}(R_{PQO}, R_{QW}) = \pi_{RQW}(R_{PQO} \Join R_{QW}) \) yields the models of \( \gamma \). \( \square \)

Lemma 3.2.3 The resolution operation over two clauses, \( (\alpha \lor Q) \) and \( (\beta \lor \neg Q) \), results in a clause \( (\alpha \lor \beta) \) for which \( models(\alpha \lor \beta) = EC_{Q'}(models(\alpha \lor Q), models(\beta \lor \neg Q)) \), where \( Q' \) is the union of scopes of both clauses excluding \( Q \). \( \square \)
The Effect of Resolution on Its Graph

\[ (\neg C) \; (A \lor B \lor C) \; (\neg A \lor B \lor E) \; (\neg B, C, E) \]

Figure 4.19: (a) The interaction graph of theory \( \varphi_1 = \{(\neg C'), (A \lor B \lor C'), (\neg A \lor B \lor E), (\neg B \lor C \lor D)\} \), and (b) the effect of resolution over \( A \) on that graph.
Directional Resolution ⇔ Adaptive Consistency

\[ \neg C \land (A \lor B \lor C) \land (\neg A \land B \lor E) \land (\neg B, C, E) \]

\[ |bucket_i| = O(\exp(w)) \]

DR time and space: \[ O(n \exp(w^*)) \]
Directional Resolution $\iff$ Adaptive Consistency

$(\sim C) (AVBVC) (\sim AvBvE)(\sim B, C, E)$

Bucket A
Bucket B
Bucket C
Bucket D
Bucket E

Input

$A \lor B \lor C \lor A \lor B \lor E$

$\sim B \lor C \lor D$

$\sim C$

Directional

Ex.

class2 828X 2019
Directional Resolution $\Leftrightarrow$ Adaptive Consistency

\[(\neg C) (AVBVC) (\neg AvBvE)(\neg B,C,E)\]

\[\left\{\begin{array}{l}
\text{Bucket A} \\
\text{Bucket B} \\
\text{Bucket C} \\
\text{Bucket D} \\
\text{Bucket E}
\end{array}\right.\]

Input:

\[A \lor B \lor C \lor A \lor B \lor E\]

\[\neg B \lor C \lor D \lor B \lor C \lor E\]

\[\neg C \lor C \lor D \lor C \lor E\]

\[D \lor E\]

\[\text{Width } w = 3\]

\[\text{Induced width } w^* = 3\]

\[|\text{bucket}_i| = O\left(\exp\left(w^*\right)\right)\]

DR time and space: $O(n \exp(\exp(w^*)))$
Directional Resolution ↔ Adaptive Consistency

Knowledge compilation

Model generation

bucket A

bucket B

bucket C

bucket D

bucket E

Input

A ∨ B ∨ C ∨ A ∨ B ∨ E

¬B ∨ C ∨ D ∨ B ∨ C ∨ E

¬C ∨ C ∨ D ∨ E ∨ D ∨ E

Directional Extension E₀

A = 0
B = 1
C = 0
D = 1
E = 0

class2 828X 2019
**Directional Resolution**

**DIRECTIONAL-RESOLUTION**

**Input:** A CNF theory $\varphi$, an ordering $d = Q_1, \ldots, Q_n$ of its variables.

**Output:** A decision of whether $\varphi$ is satisfiable. If it is, a theory $E_d(\varphi)$, equivalent to $\varphi$, else an empty directional extension.

1. **Initialize:** generate an ordered partition of clauses into buckets.
   
   $bucket_1, \ldots, bucket_n$, where $bucket_i$ contains all clauses whose highest literal is $Q_i$.

2. **for** $i \leftarrow n$ **downto** 1 process $bucket_i$:

3. **if** there is a unit clause **then** (the instantiation step)
   
   apply unit-resolution in $bucket_i$ and place the resolvents in their right buckets.
   **if** the empty clause was generated, theory is not satisfiable.

4. **else** resolve each pair $\{(\alpha \lor Q_i), (\beta \lor \neg Q_i)\} \subseteq bucket_i$.
   
   **if** $\gamma = \alpha \lor \beta$ is empty, return $E_d(\varphi) = \{\}$, theory is not satisfiable
   **else** determine the index of $\gamma$ and add it to the appropriate bucket.

5. **return** $E_d(\varphi) \leftarrow \bigcup_i bucket_i$
History

- 1960 – resolution-based Davis-Putnam algorithm

- 1962 – resolution step replaced by conditioning (Davis, Logemann and Loveland, 1962) to avoid memory explosion, resulting into a backtracking search algorithm known as Davis-Putnam (DP), or DPLL procedure.

- The dependency on induced-width was not known in 1960.

- 1994 – Directional Resolution (DR), a rediscovery of the original Davis-Putnam, identification of tractable classes (Dechter and Rish, 1994).
Properties of DR

Lemma 3.2.6 Given a theory $\varphi$ and an ordering $d = (Q_1, \ldots, Q_n)$, if $Q_i$ has at most $k$ parents in the induced graph along $d$, then the bucket of $Q_i$ in $E_d(\varphi)$ contains no more than $3^{k+1}$ clauses.

Proof: Given a clause $\alpha$ in the bucket of $Q_i$, there are three possibilities for each parent $P$ of $Q_i$: either $P$ appears in $\alpha$, $\neg P$ appears in $\alpha$, or neither of them appears in $\alpha$. Since $Q_i$ also appears in $\alpha$, either positively or negatively, the number of possible clauses in a bucket is no more than $2 \cdot 3^k < 3^{k+1}$.

Theorem 3.2.7 (complexity of DR) Given a theory $\varphi$ and an ordering of its variables $d$, the time complexity of algorithm DR along $d$ is $O(n \cdot 9^{w_d^*})$, and $E_d(\varphi)$ contains at most $n \cdot 3^{w_d^*+1}$ clauses, where $w_d^*$ is the induced width of $\varphi$’s interaction graph along $d$. \qed
Algorithms for Reasoning with graphical models

Class4
Rina Dechter