Reasoning with Deterministic and Probabilistic graphical models

Class 2: Inference in Constraint Networks

Rina Dechter
Road Map

- Graphical models
- Constraint networks Model
- Inference
- Search
- Probabilistic Networks
Example: map coloring

Variables - countries (A,B,C, etc.)
Values   - colors (red, green, blue)
Constraints:

A ≠ B, A ≠ D, D ≠ E, etc.

Constraint Networks
Constraint Satisfaction Tasks

Example: map coloring

Variables - countries (A, B, C, etc.)
Values - colors (e.g., red, green, yellow)
Constraints:

\[ A \neq B, \ A \neq D, \ D \neq E, \ etc. \]

Are the constraints consistent?

Find a solution, find all solutions

Count all solutions

Find a good (optimal) solution
A constraint network is: \( R = (X, D, C) \)

- **\( X \)** variables
- **\( D \)** domain
- **\( C \)** constraints

\[
X = \{X_1, \ldots, X_n\} \\
D = \{D_1, \ldots, D_n\}, D_i = \{v_1, \ldots, v_k\} \\
C = \{C_1, \ldots, C_i\} \\
C_i = (C_i, R_i)
\]

- **\( R \)** expresses allowed tuples over scopes

**A solution** is an assignment to all variables that satisfies all constraints (join of all relations).

**Tasks**: consistency?, one or all solutions, counting, optimization
Crossword puzzle

- Variables: x1, ..., x13
- Domains: letters
- Constraints: words from

{HOSES, LASER, SHEET, SNAIL, STEER, ALSO, EARN, HIKE, IRON, SAME, EAT, LET, RUN, SUN, TEN, YES, BE, IT, NO, US}
The Queen problem

The network has four variables, all with domains $D_i = \{1, 2, 3, 4\}$.

(a) The labeled chess board. (b) The constraints between variables.

\[
R_{12} = \{(1,3), (1,4), (2,4), (3,1), (4,1), (4,2)\}
\]
\[
R_{13} = \{(1,2), (1,4), (2,1), (2,3), (3,2), (3,4), (4,1), (4,3)\}
\]
\[
R_{14} = \{(1,2), (1,3), (2,1), (2,3), (2,4), (3,1), (3,2), (3,4), (4,2), (4,3)\}
\]
\[
R_{23} = \{(1,3), (1,4), (2,4), (3,1), (4,1), (4,2)\}
\]
\[
R_{24} = \{(1,2), (1,4), (2,1), (2,3), (3,2), (3,4), (4,1), (4,3)\}
\]
\[
R_{34} = \{(1,3), (1,4), (2,4), (3,1), (4,1), (4,2)\}
\]
Constraint’s representations

- **Relation:** allowed tuples
- **Algebraic expression:**
- **Propositional formula:**
- **Semantics:** by a relation

\[ X + Y^2 \leq 10, X \neq Y \]

\[ (a \lor b) \rightarrow \neg c \]
Partial solutions

Not all consistent instantiations are part of a solution: (a) A consistent instantiation that is not part of a solution. (b) The placement of the queens corresponding to the solution \((2, 4, 1, 3)\). (c) The placement of the queens corresponding to the solution \((3, 1, 4, 2)\).
Constraint Graphs (primal)

Queen problem
A (primal) constraint graph: a node per variable, arcs connect constrained variables. A dual constraint graph: a node per constraint’s scope, an arc connect nodes sharing variables = hypergraph.
Graph Concepts Reviews:

Primal, Hyper and Dual Graphs

- A hypergraph
- Dual graphs
- A primal graph
Propositional Satisfiability

\[ \varphi = \{ (\neg C), (A \lor B \lor C), (\neg A \lor B \lor E), (\neg B \lor C \lor D) \}. \]
Constraint graphs of 3 instances of the Radio frequency assignment problem in CELAR’s benchmark
Operations With Relations

- Intersection
- Union
- Difference
- Selection
- Projection
- Join
- Composition
Join:

Logical AND:

\[ f \land g \]
Global View of the Problem

What about counting?

Number of true tuples

Sum over all the tuples

class1 Padova
Road Map

- Graphical models
- Constraint networks Model

Inference
  - Variable elimination:
  - Tree-clustering
  - Constraint propagation

Search

Probabilistic Networks
Bucket Elimination
Adaptive Consistency (Dechter & Pearl, 1987)

Bucket E: $E \neq D$, $E \neq C$
Bucket D: $D \neq A$  $\Rightarrow D = C$
Bucket C: $C \neq B$  $\Rightarrow A \neq C$
Bucket B: $B \neq A$  $\Rightarrow B = A$
Bucket A: $A \neq C$  $\Rightarrow$ contradiction

Complexity: $O(n \exp(w^*))$

$w^*$ - induced width
The Idea of Elimination

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

Eliminate variable E ⇔ join and project
Bucket Elimination
Adaptive Consistency (Dechter & Pearl, 1987)

Bucket(E): E ≠ D, E ≠ C, E ≠ B
Bucket(D): D ≠ A || RDCB
Bucket(C): C ≠ B || RACB
Bucket(B): B ≠ A || RAB
Bucket(A): RA

Bucket(A): A ≠ D, A ≠ B
Bucket(D): D ≠ E || RDB
Bucket(C): C ≠ B, C ≠ E
Bucket(B): B ≠ E || RDBE ≠ CBE
Bucket(E): || RE

Complexity: O(n exp(w*(d))),
          w*(d) - induced width along ordering d
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 \( R_1, \ldots, R_n \in \text{bucket}_i \) do

// join all relations and "project-out" \( X_i \)

\[
R_{\text{new}} \leftarrow \prod_{(-X_i)} (\text{\# } j \in R_j)
\]

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 bucket-elimination (adaptive consistency)

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

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

- 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 ($w^*=k$).
Adaptive consistency is linear for trees and equivalent to enforcing **directional arc-consistency** (recording only unary constraints)
Tree Solving is Easy
Tree Solving is Easy

class1 Padova
Tree Solving is Easy
Tree Solving is Easy
Tree Solving is Easy

```
class1 Padova
```

```
1,2,3
```

```
1
```

```
X
```

```
<
```

```
Z
```

```
1,2
```

```
<
```

```
Y
```

```
<
```

```
1,2,3
```

```
T
```

```
1,2,3
```

```
R
```

```
1,2,3
```

```
S
```

```
1,2,3
```

```
U
```
Tree Solving is Easy
Tree Solving is Easy
Tree Solving is Easy

Constraint propagation
Solves trees in linear time
Road Map

- Graphical models
- Constraint networks Model
- **Inference**
  - Variable elimination:
    - Tree-clustering
  - Constraint propagation
- Search
- Probabilistic Networks
Tree Decomposition

Each function in a cluster

Satisfy running intersection property
Cluster Tree Elimination

1. $A B C$
   - $R(a)$, $R(b,a)$, $R(c,a,b)$
   - $BC$
   - $h_{(1,2)}(b,c) = \downarrow_a R(a) \otimes R(b,a) \otimes R(c,a,b)$

2. $B C D F$
   - $R(d,b)$, $R(f,c,d)$, $h_{(1,2)}(b,c)$
   - $BF$
   - $h_{(2,3)}(b,f) = \downarrow_{c,d} R(d,b) \otimes R(f,c,d) \otimes h_{(1,2)}(b,c)$

3. $B E F$
   - $R(e,b,f)$, $h_{(2,3)}(b,f)$

4. $E F G$
   - $R(g,e,f)$

Computes the minimal domains
CTE: Cluster Tree Elimination

Time: $O(\exp(w^*+1))$

Space: $O(\exp(sep))$

Computes the minimal domains
A tree decomposition for $R = \langle X, D, C \rangle$ is a triple $< T, \chi, \psi >$, where $T = (V,E)$ is a tree and $\chi$ and $\psi$ are labelings over vertex $v \in V$ $\chi(v) \subseteq X$ and $\psi(v) \subseteq C$ satisfying:

1. For each function $C_i \in C$ there is exactly one vertex such that $C_i \in \psi(v)$ and $\text{scope}(C_i) \subseteq \chi(v)$

2. For each variable $X_i \in X$ the set $\{v \in V | X_i \in \chi(v)\}$ forms a connected subtree (running intersection property)
Induced-width and Tree-width

Induced-width

Of ordering

Tree-width of a graph = smallest cluster in a cluster-tree
Path-width of a graph = smallest cluster in a cluster-path
Road Map

- Graphical models
- Constraint networks Model
- **Inference**
  - Variable elimination:
  - Tree-clustering
  - Constraint propagation
- **Search**
- Probabilistic Networks
Sudoku
Approximation: Constraint Propagation

Each row, column and major block must be all-different
“Well posed” if it has unique solution: 27 constraints
**Problem:** bucket-elimination/tree-clustering algorithms are intractable when induced width is large

**Approximation:** bound the size of recorded dependencies, i.e. perform local constraint propagation (local inference)
From Global to Local Consistency
A binary constraint $R(X,Y)$ is **arc-consistent** w.r.t. $X$ is every value in $x$’s domain has a match in $y$’s domain.

$$R_X = \{1,2,3\}, \ R_Y = \{1,2,3\}, \text{ constraint } X < Y$$

Only domains are reduced:

$$R_X \leftarrow \prod_X R_{XY} \bowtie D_Y$$
1 \leq X, Y, Z, T \leq 3
X < Y
Y = Z
T < Z
X \leq T

Arc-consistency
1 ≤ X, Y, Z, T ≤ 3
X < Y
Y = Z
T < Z
X ≤ T

\[ R_X \leftarrow \prod_X R_{XY} \boxdot D_Y \]
From Arc-consistency to relational arc-consistency

- **Sound**
- **Incomplete**
- **Always converges (polynomial)**
Relational Distributed Arc-Consistency

**Primal**

- A
  - 1
  - 2
  - 3
- B
  - 1
  - 2
  - 3
- C
- D
  - 1
  - 2
  - 3

**Dual**

- A
  - 1
  - 2
  - 3
- B
- C
- D
  - 1
  - 2
  - 3

class1 Padova
AC-3

AC-3(\(\mathcal{R}\))

**input:** a network of constraints \(\mathcal{R} = (X, D, C)\)

**output:** \(\mathcal{R}'\) which is the largest arc-consistent network equivalent to \(\mathcal{R}\)

1. for every pair \(\{x_i, x_j\}\) that participates in a constraint \(R_{ij} \in \mathcal{R}\)
2. \(\text{queue} \leftarrow \text{queue} \cup \{(x_i, x_j), (x_j, x_i)\}\)
3. endfor
4. while \(\text{queue} \neq \{\}\)
5. select and delete \((x_i, x_j)\) from \(\text{queue}\)
6. \(\text{Revise}((x_i), x_j)\)
7. if \(\text{Revise}((x_i), x_j)\) causes a change in \(D_i\)
8. \(\text{then } \text{queue} \leftarrow \text{queue} \cup \{(x_k, x_i), i \neq k\}\)
9. endif
10. endwhile

**Figure 3.5:** Arc-consistency-3 (AC-3)

\[O(ek^3)\]
Arc-consistency Algorithms

- **AC-1**: brute-force, distributed $O(nek^3)$
- **AC-3**: queue-based $O(ek^3)$
- **AC-4**: context-based, optimal $O(ek^2)$
- **AC-5,6,7,....** Good in special cases

**Important**: applied at every node of search

$(n$ number of variables, $e$=#constraints, $k$=domain size)


...
A pair \((x, y)\) is path-consistent relative to \(Z\), if every consistent assignment \((x, y)\) has a consistent extension to \(z\).

Figure 3.8: (a) The matching diagram of a 2-value graph coloring problem. (b) Graphical picture of path-consistency using the matching diagram.
Example: path-consistency

Figure 3.12: A graph-coloring graph (a) before path-consistency (b) after path-consistency
Path-consistency Algorithms

- Apply **Revise-3** \((O(k^3))\) until no change

\[
R_i \leftarrow R_i \cap \prod_j (R_{ij} \otimes D_{ij} \otimes R_{kj})
\]

- Path-consistency (3-consistency) adds binary constraints.
- PC-1:
- PC-2:
- PC-4 optimal:

\[
O(n^5 k^5) \quad O(n^3 k^5) \quad O(n^3 k^3)
\]
**Local i-consistency**

**i-consistency:** Any consistent assignment to any $i-1$ variables is consistent with at least one value of any $i$-th variable

---

**ARC-CONSISTENCY**

---

**PATH-CONSISTENCY**

---

**i-CONSISTENCY**

---

**Figure 3.17:** The scope of consistency enforcing: (a) arc-consistency, (b) path-consistency, (c) i-consistency
Linear inequalities

\[ x + y + z \leq 15, \ z \geq 13 \implies \]

Boolean constraint propagation, unit resolution

\[ x \leq 2, \ y \leq 2 \]

\[ (A \lor B \lor \neg C), (\neg B) \implies \]

\[ (A \lor \neg C) \]
Directional Resolution $\leftrightarrow$ Adaptive Consistency

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

DR time and space: $O(n \exp(w^*))$
Directional i-consistency

Adaptive

$R_{DCB}$

$d$-path

$R_{DC}, R_{DB}$

$R_{CB}$

$d$-arc

$R_D$

$R_C$

$R_D$
Greedy Algorithms for Induced-Width

- Min-width ordering
- Min-induced-width ordering
- Max-cardinality ordering
- Min-fill ordering
- Chordal graphs
- Hypergraph partitionings

(Project: present papers on induced-width, run algorithms for induced-width on new benchmarks...
Min-width ordering

**Proposition:** algorithm min-width finds a min-width ordering of a graph

**Complexity:** ?

$O(e)$
**min-induced-width (miw)**

*input: a graph \( G = (V; E), V = \{v_1; \ldots; v_n\} \)
*output: A miw ordering of the nodes \( d = (v_1; \ldots; v_n) \).

1. for \( j = n \) to 1 by -1 do
2. \( r \leftarrow \) a node in \( V \) with smallest degree.
3. put \( r \) in position \( j \).
4. connect \( r \)'s neighbors: \( E \leftarrow E \cup \{(v_i; v_j) | (v_i; r) \in E; (v_j; r) \in E\} \),
5. remove \( r \) from the resulting graph: \( V \leftarrow V - \{r\} \).

**min-fill (min-fill)**

*input: a graph \( G = (V; E), V = \{v_1; \ldots; v_n\} \)
*output: An ordering of the nodes \( d = (v_1; \ldots; v_n) \).

1. for \( j = n \) to 1 by -1 do
2. \( r \leftarrow \) a node in \( V \) with smallest fill edges for his parents.
3. put \( r \) in position \( j \).
4. connect \( r \)'s neighbors: \( E \leftarrow E \cup \{(v_i; v_j) | (v_i; r) \in E; (v_j; r) \in E\} \),
5. remove \( r \) from the resulting graph: \( V \leftarrow V - \{r\} \).

**Theorem:** A graph is a tree iff it has both width and induced-width of 1.
Example

We see again that $G$ in the Figure (a) is not chordal since the parents of $A$ are not connected in the max-cardinality ordering in Figure (d). If we connect $B$ and $C$, the resulting induced graph is chordal.
Cordial Graphs; Max-Cardinality Ordering

- A graph is cordal if every cycle of length at least 4 has a chord.
- Finding $w^*$ over chordal graph is easy using the max-cardinality ordering.
- If $G^*$ is an induced graph it is chordal.
- $K$-trees are special chordal graphs.
- Finding the max-clique in chordal graphs is easy (just enumerate all cliques in a max-cardinality ordering.)
Max-cardinality ordering
Graph Concepts: Hypergraphs and Dual Graphs

- **A hypergraph** is $H = (V,S)$, $V = \{v_1, \ldots, v_n\}$ and a set of subsets **Hyperedges**: $S = \{S_1, \ldots, S_l\}$.

- **Dual graphs** of a hypergraph: The nodes are the hyperedges and a pair of nodes is connected if they share vertices in $V$. The arc is labeled by the shared vertices.

- **A primal (Markov, moral) graph** of a hypergraph $H = (V,S)$ has $V$ as its nodes, and any two nodes are connected by an arc if they appear in the same hyperedge.

- **Factor Graphs**.
Which greedy algorithm is best?

- **MinFill**, prefers a node who add the least number of fill-in arcs.

- *Empirically, fill-in is the best among the greedy algorithms (MW, MIW, MF, MC)*

- **Complexity of greedy orderings?**
  - **MW** is \( O(?) \), **MIW**: \( O(?) \) **MF** (?) **MC** is \( O(mn) \)
Road Map

- **Graphical models**
- **Constraint networks Model**
- **Inference**
  - Variable elimination:
  - Tree-clustering
  - Constraint propagation
- **Search**
- **Probabilistic Networks**