Adversial search

Resource Limit : ① Depth-limited ② Limiting branching factor

Evaluation function : $Eval(s) = w_1 f_1(s) + w_2 f_2(s) + \cdots$

Minimax

```
def value(state):
    if the state is a terminal state: return the state's utility
    if the next agent is MAX: return max-value(state)
    if the next agent is MIN: return min-value(state)
```

```
def max-value(state):
    initialize v = -∞
    for each successor of state:
        v = max(v, value(successor))
    return v
```

```
def min-value(state):
    initialize v = +∞
    for each successor of state:
        v = min(v, value(successor))
    return v
```

α: MAX's best option on path to root
β: MIN's best option on path to root

Alpha-beta

```
def max-value(state, α, β):
    initialize v = -∞
    for each successor of state:
        v = max(v, value(successor, α, β))
        if v ≥ β return v
        α = max(α, v)
    return v
```

```
def min-value(state , α, β):
    initialize v = +∞
    for each successor of state:
        v = min(v, value(successor, α, β))
        if v ≤ α return v
        β = min(β, v)
    return v
```

Expectimax

no pruning!

```
def value(state):
    if the state is a terminal state: return the state's utility
    if the next agent is MAX: return max-value(state)
    if the next agent is EXP: return exp-value(state)
```

```
def max-value(state):
    initialize v = -∞
    for each successor of state:
        v = max(v, value(successor))
    return v
```

```
def exp-value(state):
    initialize v = 0
    for each successor of state:
        p = probability(successor)
        v += p * value(successor)
    return v
```

Propositional Logic

Syntax ↔ semantic

conjunctive : $\wedge$
disjunctive : $\vee$

Valid / satisfiable / un-

$(\alpha \wedge \beta) \equiv (\beta \wedge \alpha)$ commutativity of $\wedge$

$(\alpha \vee \beta) \equiv (\beta \vee \alpha)$ commutativity of $\vee$

$((\alpha \wedge \beta) \wedge \gamma) \equiv (\alpha \wedge (\beta \wedge \gamma))$ associativity of $\wedge$

$((\alpha \vee \beta) \vee \gamma) \equiv (\alpha \vee (\beta \vee \gamma))$ associativity of $\vee$

$\neg(\neg\alpha) \equiv \alpha$ double-negation elimination

$(\alpha \Rightarrow \beta) \equiv (\neg\beta \Rightarrow \neg\alpha)$ contraposition

$(\alpha \Rightarrow \beta) \equiv (\neg\alpha \vee \beta)$ implication elimination

$(\alpha \Leftrightarrow \beta) \equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha))$ biconditional elimination

$\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta)$ de Morgan

$\neg(\alpha \vee \beta) \equiv (\neg\alpha \wedge \neg\beta)$ de Morgan

$(\alpha \wedge (\beta \vee \gamma)) \equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma))$ distributivity of $\wedge$ over $\vee$

$(\alpha \vee (\beta \wedge \gamma)) \equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma))$ distributivity of $\vee$ over $\wedge$

Entailment : $\alpha \models \beta$ $model(\alpha) \subseteq model(\beta)$

Proof : $\alpha \vdash \beta$ { model checking ✗ / rule of inference }

Sound : $\alpha \vdash \beta \rightarrow \alpha \models \beta$

complete : $\alpha \models \beta \rightarrow \alpha \vdash \beta$

$B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$ **Convert to CNF**

1.Eliminate $\Leftrightarrow$, replacing $\alpha \Leftrightarrow \beta$ with $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$.

$(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$

2.Eliminate $\Rightarrow$, replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \vee \beta$.

$(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg(P_{1,2} \vee P_{2,1}) \vee B_{1,1})$

3.Move $\neg$ inwards using de Morgan's rules and double-negation:

$(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge ((\neg P_{1,2} \wedge \neg P_{2,1}) \vee B_{1,1})$

4.Apply distributivity law ($\wedge$ over $\vee$) and flatten:

$(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1})$

Prove $KB \models \alpha$ → show $KB \wedge \neg\alpha$ is unsatisfiable

Horn Logic 1° $P_1 \wedge P_2 \wedge \cdots \wedge P_n \Rightarrow Q$, 2° $\neg P_1 \vee \neg P_2 \vee \cdots \vee \neg P_n \vee Q$

⇒ Forward chain : Add new clauses into KB until Q is added ⇒ data-driving

Backward chain : goal-driving
\# avoid loop (check if new subgoal is on the goal stack)
\# avoid repeated work (has already be proved or fail)

First-Order Logic

Syntax Logical symbols: Connectives (¬, ⇒, ∧, ∨, ⇔) Quantifiers (∀, ∃) Variable (x, y) Equality (=)

Non-logical symbols: Constants (King, 2) Predicates (Brother, >) Function (Sqrt, LeftLegOf)

Term: constant / variable / function(term1, term2, ...)

Atomatic sentence: predicate(term1, term2, ...) / term1 = term2.

Sibling(A,B) ⇒ Sibling(B,A)

Complex sentence: automatic sentences using connectives (¬S, S1 ⇒ S2, ¬(1,2) ∨ ≤(1,2))

Semantic model = objects + relations + interpretation (specify symbols)

$\forall x\ At(x, STU) \Rightarrow Smart(x)$ $\exists x\ At(x, STU) \land Smart(x)$

$\forall x \forall y$ same as $\forall y \forall x$, $\exists x \exists y$ same as $\exists y \exists x$, $\exists x \forall y$ not $\forall y \exists x$.

$\forall x\ P(x) \equiv \neg \exists x \neg P(x)$

$\exists x\ P(x) \equiv \neg \forall x \neg P(x)$

In FOL, every var must be bound. $\forall x\ P(x,y)$ is not valid.

Inference. I) Propositionalization (UI | EI)

II) Unification substitude. MGU = {y/John, x/z}

III) FC/BC (Horn Logic) ← universally quantified.

IV) Resolution

1. Eliminate biconditionals and implications
   ∀x [¬∀y ¬*Animal*(y) ∨ *Loves*(x,y)] ∨ [∃y *Loves*(y,x)]

2. Move ¬ inwards: ¬∀x p ≡ ∃x ¬p, ¬∃x p ≡ ∀x ¬p
   ∀x [∃y ¬(¬Animal(y) ∨ Loves(x,y))] ∨ [∃y Loves(y,x)]
   ∀x [∃y ¬¬Animal(y) ∧ ¬Loves(x,y)] ∨ [∃y Loves(y,x)]
   ∀x [∃y Animal(y) ∧ ¬Loves(x,y)] ∨ [∃y Loves(y,x)]

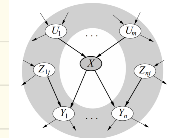
3. Standardize variables: each quantifier should use different variable
   ∀x [∃y Animal(y) ∧ ¬Loves(x,y)] ∨ [∃z Loves(z,x)]

4. Skolemize: a more general form of existential instantiation. Each existential variable is replaced by a Skolem function of the enclosing universally quantified variables:
   ∀x [Animal(F(x)) ∧ ¬Loves(x,F(x))] ∨ Loves(G(x),x)

5. Drop universal quantifiers:
   [Animal(F(x)) ∧ ¬Loves(x,F(x))] ∨ Loves(G(x),x)

6. Distribute ∨ over ∧ :
   [Animal(F(x)) ∨ Loves(G(x),x)] ∧ [¬Loves(x,F(x)) ∨ Loves(G(x),x)]

Bayesian Network

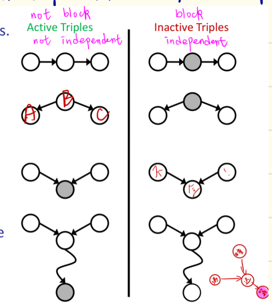


Syntax DAG + CPTs

Semantic

Markov Blanket: independent of all other vars given its parents, children, children's par

- Question: X, Y, Z are non-intersecting subsets of nodes. Are X and Y conditionally independent given Z?
- A triple is active in the following three cases
  - Causal chain A → B → C where B is unobserved (either direction)
  - Common cause A ← B → C where B is unobserved
  - Common effect (aka v-structure)
    A → B ← C where B *or one of its descendents* is observed
- A path is active if each triple along the path is active
- A path is blocked if it contains a single inactive triple
- If all (undirected) paths from X to Y are blocked, then X is said to be **"d-separated"** from Y by Z
- If d-separated, then X and Y are conditionally independent given Z



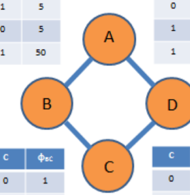
Markov Networks Undirected graph + potentials

A	B	C	D	$\phi_{AB}\phi_{BC}\phi_{CD}\phi_{AD}$
0	0	0	0	250
0	0	0	1	37500
0	0	1	0	50000
0	0	1	1	625000
0	1	0	0	1125
0	1	0	1	168750
0	1	1	0	50000
0	1	1	1	625000
1	0	0	0	250
1	0	0	1	375
1	0	1	0	50000
1	0	1	1	6250
1	1	0	0	112500
1	1	0	1	168750
1	1	1	0	5000000
1	1	1	1	625000

Z = 7520750

A	B	ϕ_{AB}
0	0	50
0	1	5
1	0	5
1	1	50

A	D	ϕ_{AD}
0	0	5
0	1	50
1	0	50
1	1	5



B	C	ϕ_{BC}
0	0	1
0	1	5
1	0	45
1	1	50

C	D	ϕ_{CD}
0	0	1
0	1	15
1	0	40
1	1	50

Exact Inference

Enumeration exponential

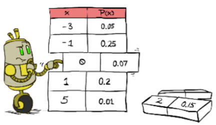
- General case:
  - Evidence variables: $E_1 \dots E_k = e_1 \dots e_k$
  - Query variable: Q
  - Hidden variables: $H_1 \dots H_r$

  $\left.\right\}$ $X_1, X_2, \dots X_n$ *All variables*

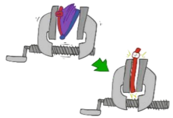
- We want:

$$P(Q|e_1 \dots e_k)$$

- Step 1: Select the entries consistent with the evidence



- Step 2: Sum out H to get joint of Query and evidence



$$P(Q, e_1 \dots e_k) = \sum_{h_1 \dots h_r} \underbrace{P(Q, h_1 \dots h_r, e_1 \dots e_k)}_{X_1, X_2, \dots X_n}$$

- Step 3: Normalize

$$\times \frac{1}{Z}$$

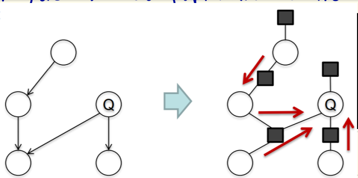
$$Z = \sum_q P(Q, e_1 \cdots e_k)$$

$$P(Q|e_1 \cdots e_k) = \frac{1}{Z} P(Q, e_1 \cdots e_k)$$

Variable Elimination exponential, often better

$\sum_t \sum_r P(L|t)P(r)P(t|r) \Rightarrow \sum_t P(L|t) \sum_r P(r)P(t|r)$

Polytree: Directed graph without undirected cycles ⇒ linear complexity



Bayesian Network Factor Graph

- Input: evidence $e_1,..,e_k$
- For i=1, 2, …, n
  - Sample X_i from $P(X_i \mid parents(X_i))$
  - If x_i not consistent with evidence
    - Reject: Return, and no sample is generated in this cycle
- Return $(x_1, x_2, \dots, x_n)$

reject

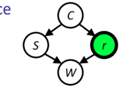
- For i=1, 2, …, n (in topological order)
  - Sample X_i from $P(X_i \mid parents(X_i))$
- Return $(x_1, x_2, \dots, x_n)$

prior

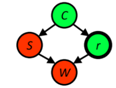
- Input: evidence $e_1,..,e_k$
- $w = 1.0$
- for i=1, 2, …, n
  - if X_i is an evidence variable
    - x_i = observed value$_i$ for X_i
    - Set $w = w * P(x_i \mid Parents(X_i))$
  - else
    - Sample x_i from $P(X_i \mid Parents(X_i))$
- return $(x_1, x_2, \dots, x_n)$, w

likelihood

- Step 1: Fix evidence
  - R = true

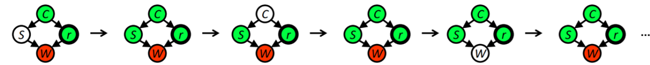


- Step 2: Initialize other variables
  - Randomly

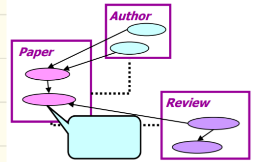


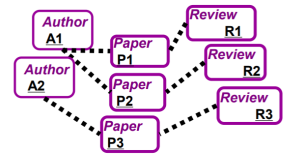
Gibbs

- Step 3: Repeat
  - Choose an arbitrary non-evidence variable X
  - Resample X from $P(X \mid markov\_blanket(X))$



Sample $S \sim P(S \mid c, r, \neg w)$ Sample $C \sim P(C \mid s, r)$ Sample $W \sim P(W \mid s, r)$





PRM (S, Θ) + relational skeleton (σ) =

probability distribution over instantiations of attributes I:

$$P(I \mid \sigma, S, \Theta) = \prod_{x \in \sigma} \prod_{x.A} P(x.A \mid parents_{S,\sigma}(x.A))$$

Objects Attributes

Search Problem

State Space [location (x) dots booleans)] / Successor function / Start & Goal state

World state: Agent pos × $2^{\text{Food counts}}$ × Ghost $\text{pos}^{\#\text{ of Ghosts}}$ × actions

Tree Search

		Strategy	Fringe	is Complete	is Optimal	Time complexity
Uninformed	DFS	a deepest node	LIFO stack	Yes | no cycle	No | leftmost	$O(b^m)$
	BFS	a shallowest node	FIFO queue	Yes	Yes | same costs	$O(b^s)$
	UCS	a cheapest node	priority queue	Yes	Yes	$O(b^{C^*/\varepsilon})$
Informed	Greedy	a node that is closest to a goal		badly guide DFS		
	A*	orders by $f(n) = g(n) + h(n)$	admissible (optimistic): $0 \le h(n) \le h^*(n)$		Yes	

path cost / backward cost $g(n)$

goal proximity / forward cost $h(n)$

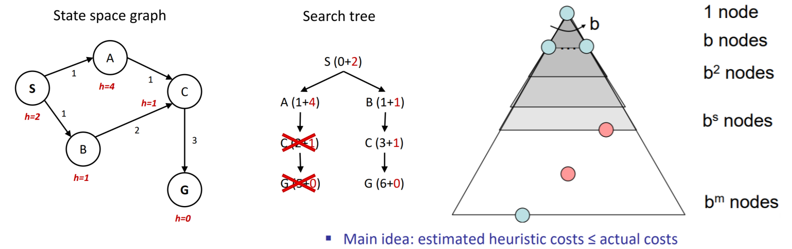
C^*: solution costs C^*

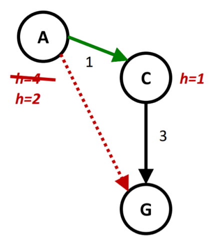
arcs cost at least ε

Heuristic

A function that estimates how close a state is to a goal

Graph Search never expand a node twice   A* with consistent heuristic





- Main idea: estimated heuristic costs ≤ actual costs
  - Admissibility: heuristic cost ≤ actual cost to goal
    - h(A) ≤ actual cost from A to G
  - Consistency: heuristic "arc" cost ≤ actual cost for each arc
    - h(A) – h(C) ≤ cost(A to C)
  - Consistency implies admissibility
- Consequences of consistency:
  - The f value along a path never decreases, because:
    - h(A) ≤ cost(A to C) + h(C)

Constraint Satisfication Problems

Variables / Domains / constraints

Basic solution: Backtracking ① One variable at a time ② check constraints as you go

= DFS + variable-ordering + Fail-on-violation

Filtering: Forward Checking: Cross off values that violates a constraint when added to assignment; whenever any variables has no value left, backtrack.

Arc Consistency: An arc X→Y is consistent iff for every x in the tail there is some y in the head could be assigned without voilating a constraint.

* Delete from the tail * If Y loses a value, arc X→Y needs to be re-checked.

Ordering: Minimum Remaing Value: the variable with the fewest legal left values in domain

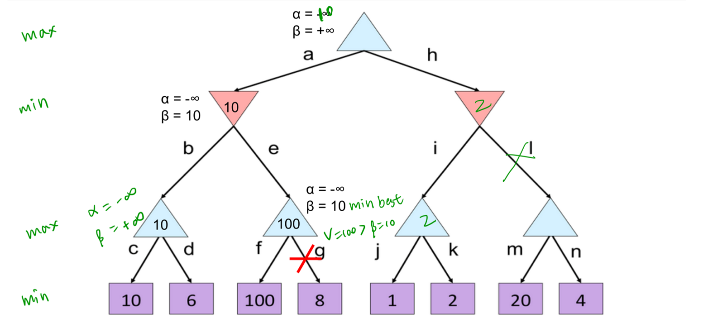
Least Constraining Value: the one rules out the fewest values in the remaing vars.

Structure: Tree structure: Choose a root variable, order varible that parents precede childs.

Remove backword: n→2, remove inconsistent + Assign forward: 1→n assign x

Cutset structure: a set of variables s.t. the remaining constraint is a tree.

Alpha-Beta Example 2



Adversial Search

Resource Limit: ① Depth-limited ② Limiting branching factor

Evaluation function: $Eval(s) = w_1 f_1(s) + w_2 f_2(s) + \cdots$

Minimax

```
def value(state):
    if the state is a terminal state: return the state's utility
    if the next agent is MAX: return max-value(state)
    if the next agent is MIN: return min-value(state)

def max-value(state):
    initialize v = -∞
    for each successor of state:
        v = max(v, value(successor))
    return v

def min-value(state):
    initialize v = +∞
    for each successor of state:
        v = min(v, value(successor))
    return v
```

α: MAX's best option on path to root
β: MIN's best option on path to root

Alpha-beta

```
def max-value(state, α, β):
    initialize v = -∞
    for each successor of state:
        v = max(v, value(successor, α, β))
        if v ≥ β return v
        α = max(α, v)
    return v

def min-value(state , α, β):
    initialize v = +∞
    for each successor of state:
        v = min(v, value(successor, α, β))
        if v ≤ α return v
        β = min(β, v)
    return v
```

Expectimax

no pruning!

```
def value(state):
    if the state is a terminal state: return the state's utility
    if the next agent is MAX: return max-value(state)
    if the next agent is EXP: return exp-value(state)

def max-value(state):
    initialize v = -∞
    for each successor of state:
        v = max(v, value(successor))
    return v

def exp-value(state):
    initialize v = 0
    for each successor of state:
        p = probability(successor)
        v += p * value(successor)
    return v
```

Propositional Logic

Syntax & semantic   {conjunctive: ∧ ; disjunctive: ∨}   Valid / satisfiable / un-

$(\alpha \land \beta) \equiv (\beta \land \alpha)$ commutativity of $\land$

$(\alpha \lor \beta) \equiv (\beta \lor \alpha)$ commutativity of $\lor$

$((\alpha \land \beta) \land \gamma) \equiv (\alpha \land (\beta \land \gamma))$ associativity of $\land$

$((\alpha \lor \beta) \lor \gamma) \equiv (\alpha \lor (\beta \lor \gamma))$ associativity of $\lor$

$\neg(\neg\alpha) \equiv \alpha$ double-negation elimination

$(\alpha \Rightarrow \beta) \equiv (\neg\beta \Rightarrow \neg\alpha)$ contraposition

$(\alpha \Rightarrow \beta) \equiv (\neg\alpha \lor \beta)$ implication elimination

$(\alpha \Leftrightarrow \beta) \equiv ((\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha))$ biconditional elimination

$\neg(\alpha \land \beta) \equiv (\neg\alpha \lor \neg\beta)$ de Morgan

$\neg(\alpha \lor \beta) \equiv (\neg\alpha \land \neg\beta)$ de Morgan

$(\alpha \land (\beta \lor \gamma)) \equiv ((\alpha \land \beta) \lor (\alpha \land \gamma))$ distributivity of $\land$ over $\lor$

$(\alpha \lor (\beta \land \gamma)) \equiv ((\alpha \lor \beta) \land (\alpha \lor \gamma))$ distributivity of $\lor$ over $\land$

Entailment: $\alpha \models \beta$ $model(\alpha) \subseteq model(\beta)$

Proof: $\alpha \vdash \beta$ {model checking × ; rule of inference}

Sound: $\alpha \vdash \beta \rightarrow \alpha \models \beta$

complete: $\alpha \models \beta \rightarrow \alpha \vdash \beta$

$B_{1,1} \Leftrightarrow (P_{1,2} \lor P_{2,1})$ Convert to CNF

1. Eliminate ⇔, replacing α ⇔ β with (α ⇒ β)∧(β ⇒ α).
   $(B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land ((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1})$
2. Eliminate ⇒, replacing α ⇒ β with ¬α∨ β.
   $(\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg(P_{1,2} \lor P_{2,1}) \lor B_{1,1})$
3. Move ¬ inwards using de Morgan's rules and double-negation:
   $(\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land ((\neg P_{1,2} \land \neg P_{2,1}) \lor B_{1,1})$
4. Apply distributivity law (∧ over ∨) and flatten:
   $(\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg P_{1,2} \lor B_{1,1}) \land (\neg P_{2,1} \lor B_{1,1})$

Prove $KB \models \alpha$ → show $KB \land \neg\alpha$ is unsatisfiable

Horn Logic 1° $P_1 \land P_2 \land \cdots \land P_n \Rightarrow Q$, 2° $\neg P_1 \lor \neg P_2 \lor \cdots \lor \neg P_n \lor Q$

⇒ Forward chain: Add new clauses into KB until Q is added ⇒ data-driving

Backward chain: goal-driving # avoid loop (check if new subgoal is on the goal stack)

avoid repeated work (has already be proved or fail)

First-Order Logic

Syntax Logical symbols: Connectives ($\neg, \Rightarrow, \wedge, \vee, \Leftrightarrow$) Quantifiers ($\forall, \exists$) Variable ($x, y$) Equality ($=$)

Non-logical symbols: Constants (King, 2) Predicates (Brother, >) Function (Sqrt, LeftLegOf)

Term: constant / variable / function (term1, term2, ...)

Atomatic sentence: predicate (term1, term2, ...) / term1 = term2.

Complex sentence: automatic sentences using connectives ($\neg S$, $S_1 \Rightarrow S_2$, $>(1,2) \vee \leq(1,2)$) Sibling(A,B) ⇒ Sibling(B,A))

Semantic model = objects + relations + interpretation (specify symbols)

$\forall x\ At(x, STU) \Rightarrow Smart(x)$ $\quad \exists x\ At(x, STU) \wedge Smart(x)$

$\forall x \forall y$ same as $\forall y \forall x$, $\exists x \exists y$ same as $\exists y \exists x$, $\exists x \forall y$ not $\forall y \exists x$.

$\forall x\ P(x) \equiv \neg \exists x \neg P(x)$

$\exists x\ P(x) \equiv \neg \forall x \neg P(x)$

In FOL, every var must be bound. $\forall x\ P(x, y)$ is not valid.

Inference. I) Propositionalization (UI / EI)

II) Unification substitude. MGU = {y/John, x/z}

III) FC / BC (Horn Logic) ⟸ universally quantified.

IV) Resolution

1. Eliminate biconditionals and implications
   ∀x [¬∀y ¬*Animal(y)* ∨ *Loves(x,y)*] ∨ [∃y *Loves(y,x)*]
2. Move ¬ inwards: ¬∀x p ≡ ∃x ¬p, ¬∃x p ≡ ∀x ¬p
   ∀x [∃y ¬(¬Animal(y) ∨ Loves(x,y))] ∨ [∃y Loves(y,x)]
   ∀x [∃y ¬¬Animal(y) ∧ ¬Loves(x,y)] ∨ [∃y Loves(y,x)]
   ∀x [∃y Animal(y) ∧ ¬Loves(x,y)] ∨ [∃y Loves(y,x)]
3. Standardize variables: each quantifier should use different variable
   ∀x [∃y Animal(y) ∧ ¬Loves(x,y)] ∨ [∃z Loves(z,x)]
4. Skolemize: a more general form of existential instantiation. Each existential variable is replaced by a Skolem function of the enclosing universally quantified variables:
   ∀x [Animal(F(x)) ∧ ¬Loves(x,F(x))] ∨ Loves(G(x),x)
5. Drop universal quantifiers:
   [Animal(F(x)) ∧ ¬Loves(x,F(x))] ∨ Loves(G(x),x)
6. Distribute ∨ over ∧ :
   [Animal(F(x)) ∨ Loves(G(x),x)] ∧ [¬Loves(x,F(x)) ∨ Loves(G(x),x)]

Exact Inference

Enumeration exponential

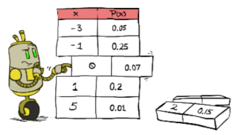
- General case:
  - Evidence variables: $E_1 \dots E_k = e_1 \dots e_k$
  - Query variable: Q
  - Hidden variables: $H_1 \dots H_r$

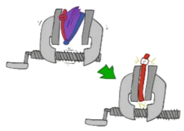
  $\} X_1, X_2, \dots X_n$ *All variables*

- We want:

$$P(Q|e_1 \dots e_k)$$

- Step 1: Select the entries consistent with the evidence
- Step 2: Sum out H to get joint of Query and evidence
- Step 3: Normalize





$$P(Q, e_1 \dots e_k) = \sum_{h_1 \dots h_r} \underbrace{P(Q, h_1 \dots h_r, e_1 \dots e_k)}_{X_1, X_2, \dots X_n}$$

$$\times \frac{1}{Z}$$

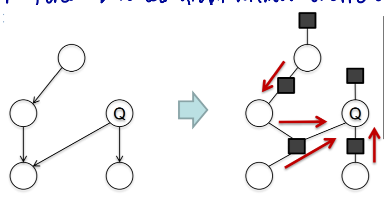
$$Z = \sum_q P(Q, e_1 \cdots e_k)$$

$$P(Q|e_1 \cdots e_k) = \frac{1}{Z} P(Q, e_1 \cdots e_k)$$

Variable Elimination exponential often better

$\sum_t \sum_r P(L|t) P(r) P(t|r) \Rightarrow \sum_t P(L|t) \sum_r P(r) P(t|r)$

Polytree: Directed graph without undirected cycles ⇒ linear complexity



Bayesian Network $\qquad$ Factor Graph

- Input: evidence $e_1, .., e_k$
- For i=1, 2, ..., n
  - Sample x_i from $P(X_i \mid parents(X_i))$
  - If x_i not consistent with evidence
    - Reject: Return, and no sample is generated in this cycle
- Return $(x_1, x_2, \dots, x_n)$

reject

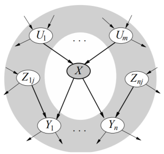
- For i=1, 2, ..., n (in topological order)
  - Sample X_i from $P(X_i \mid parents(X_i))$
- Return $(x_1, x_2, \dots, x_n)$

prior

- Input: evidence $e_1, .., e_k$
- $w = 1.0$
- for i=1, 2, ..., n
  - if X_i is an evidence variable
    - x_i = observed value$_i$ for X_i
    - Set w = w * $P(x_i \mid Parents(X_i))$
  - else
    - Sample x_i from $P(X_i \mid Parents(X_i))$
- return $(x_1, x_2, \dots, x_n)$, w

likelihood

Bayesian Network

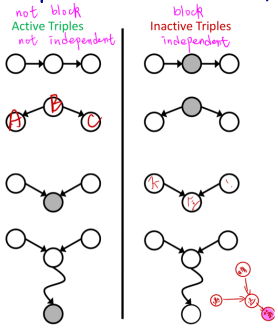


Syntax DAG + CPTs

Semantic

Markov Blanket: indepent of all other vars given its parents, children, children's par

- Question: X, Y, Z are non-intersecting subsets of nodes. Are X and Y conditionally independent given Z?
- A triple is active in the following three cases
  - Causal chain A → B → C where B is unobserved (either direction)
  - Common cause A ← B → C where B is unobserved
  - Common effect (aka v-structure) A → B ← C where B *or one of its descendents* is observed
- A path is active if each triple along the path is active
- A path is blocked if it contains a single inactive triple
- If all undirected paths from X to Y are blocked, then X is said to be "**d-separated**" from Y by Z
- If d-separated, then X and Y are conditionally independent given Z



Markov Networks

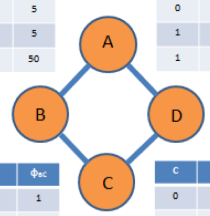
Undirected graph + potentials

A	B	C	D	$\phi_{AB}\phi_{BC}\phi_{CD}\phi_{AD}$
0	0	0	0	250
0	0	0	1	37500
0	0	1	0	50000
0	0	1	1	625000
0	1	0	0	1125
0	1	0	1	168750
0	1	1	0	50000
0	1	1	1	625000
1	0	0	0	250
1	0	0	1	375
1	0	1	0	50000
1	0	1	1	6250
1	1	0	0	112500
1	1	0	1	168750
1	1	1	0	5000000
1	1	1	1	625000

Z = 7520750

A	B	ϕ_{AB}
0	0	50
0	1	5
1	0	5
1	1	50

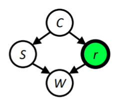
A	D	ϕ_{AD}
0	0	5
0	1	50
1	0	50
1	1	5



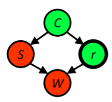
B	C	ϕ_{BC}
0	0	1
0	1	5
1	0	45
1	1	50

C	D	ϕ_{CD}
0	0	1
0	1	15
1	0	40
1	1	50

- Step 1: Fix evidence
  - *R* = true

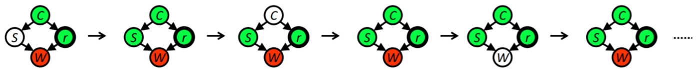


- Step 2: Initialize other variables
  - Randomly

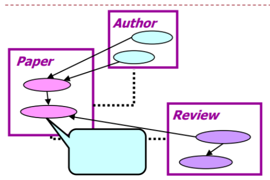


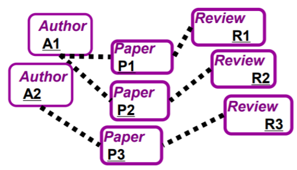
Gibbs

- Step 3: Repeat
  - Choose an arbitrary non-evidence variable *X*
  - Resample *X* from *P(X | markov_blanket(X))*



Sample $S \sim P(S \mid c, r, \neg w)$ $\qquad$ Sample $C \sim P(C \mid s, r)$ $\qquad$ Sample $W \sim P(W \mid s, r)$





PRM (S, Θ) + relational skeleton (σ) =

probability distribution over instantiations of attributes I:

$$P(I \mid \sigma, S, \Theta) = \prod_{x \in \sigma} \prod_{x.A} P(x.A \mid parents_{S,\sigma}(x.A))$$

Objects $\quad$ Attributes

搜索

- 定义
  - State Space, Successor function, 起止条件
  - Completeness, Optimality
    - complete: 一定能找到一个解决方案
    - Optimality: 一定能找到最好的解决方案
- Tree Search
  - 全局搜索
    - DFS
    - BFS
    - UCS
  - 启发式搜索
    - Heuristic, admissible heuristic
    - Greedy, A*
      - Greedy 不是optimal
- 图搜索
  - A* with consistent heuristic
    - Complete and Optimal

贝叶斯网络

- 语法:
  - 有向无环图(DAG) + 条件概率表(CPTs)
- 语义:
  - Global semantics 所有变量的 joint 分布等于所有的 CPT 相乘
  - 条件独立关系
  - D-separation （路径是有向还是无向?
- 马尔科夫网络
  - 无向图 + potentials
  - Semantics
    - 一组值的概率
    - 条件独立
- 马尔科夫网络推理
  - 精确推理
    - 枚举法
    - 变量消元
      - join + sum out
    - Polytrees
- 大致推理
  - Prior Sampling
  - Rejection Sampling
  - Likelihood Weighting
  - Gibbs Sampling

命题逻辑

- 表示
  - 语法
    - 逻辑符号
    - 命题符号
  - 语义
    - 每一个 model 都对于每一个命题符号赋予一个 T 或者 F
    - 对每一个 model 可以得出一个语句的 T 或者 F
- Inference
  - CNF
  - 最终推出一个为空的 domain
- Concepts
  - Validity 永远为真
  - Satisfiability
  - entailment 蕴含
  - proof 对蕴含的证明
  - soundness
  - completeness
- Horn 逻辑
  - 表示
    - CNF -> Q
  - 推理
    - Modus Ponens 推理规则
      - 如果左侧的 P 都在 KB 里了，那么Q也可以加入到 KB 里
    - Forward
    - Backward

敌对搜索

- Game-tree, Minimax
- 三种节点
  - 叶节点是游戏结束的节点（有分数）
  - Max 节点是孩子节点中最大的
  - Min 节点是孩子节点中最小的
- 实际中不可能到叶节点 (时间复杂度太高)
  - 优化方法：对每一个 state 打分
- 剪枝(alpha-beta)
  - alpha: 根节点到当前节点的路径上，Max最好的选择
  - beta: 根节点到当前节点上，Min最好的选择
- ExpectMax

CSP: Constraint Satisfaction Problems

- 一类特殊的搜索问题
- Basic Solution: Backtracking search
  - 本质上是深搜
- 三个加速方式：
  - Filtering
    - Forward Checking
    - Arc Consistency
  - Ordering
    - Minimum Remaining Values, Least Constraining Value
  - Structure
    - 树结构，Cutset conditioning
- Local Search
  - 爬山法（不是 complete 的）
  - 有可能陷入局部最优

一阶（谓词）逻辑

- 语法:
  - Constant, Predicate, Function, Variable, Connective, Quantifier, equality
- 语义:
  - 一个 Model 包括: Objects, relations, interpretation
- 推理:
  - Propositionalization
    - UI, EI
  - Unification 替换
  - Forward/Backward Chaining
  - Resolution

概率逻辑

- Probabilistic Relational Models (PRM)
  - 模板
    - 一个物体的属性可以依赖于与他相关的属性
- Markov Logic
- 一阶逻辑 + 权重
  - 权重越大表示这个表达式越需要满足
- 每一个一阶逻辑对应到一个马尔科夫网络的 clique

Problem

Space [location (dots booleans)] / Successor function / Start & Goal state

State: Agent pos $\times 2^{\text{Food counts}} \times$ Ghost pos$^{\#\text{ of Ghosts}}$ $\times$ actions

Search

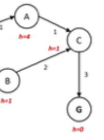
	Strategy	Fringe	is Complete	is Optimal	Time Complexity
DFS	a deepest node	LIFO stack	Yes | no cycle	No | leftmost	$O(b^m)$
BFS	a shallowest node	FIFO queue	Yes	Yes | same costs	$O(b^s)$
UCS	a cheapest node (path cost / backward cost $g(n)$)	priority queue	Yes	Yes	$O(b^{C^*/\varepsilon})$
Greedy	a node that is closest to a goal (goal proximity / forward cost $h(n)$)		badly guide DFS		
A*	orders by $f(n) = g(n) + h(n)$	Admissible (optimistic): $0 \le h(n) \le h^*(n)$		Yes	

C^*: solution costs C^*, arcs cost at least ε

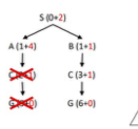
Heuristic
A function that estimates how close a state is to a goal

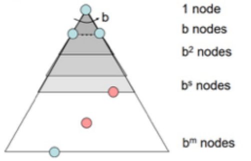
Graph Search: never expand a node twice. A* with consistent heuristic

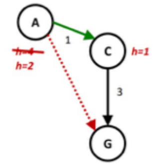
State space graph



Search tree







- Main idea: estimated heuristic costs ≤ actual costs
  - Admissibility: heuristic cost ≤ actual cost to goal
    - h(A) ≤ actual cost from A to G
  - Consistency: heuristic "arc" cost ≤ actual cost for each arc
    - h(A) – h(C) ≤ cost(A to C)
  - Consistency implies admissibility
- Consequences of consistency:
  - The f value along a path never decreases, because:
    - h(A) ≤ cost(A to C) + h(C)

Constraint Satisfaction Problems

Variables / Domains / constraints

Basic solution: Backtracking ① One variable at a time ② check constraints as you go = DFS + variable-ordering + fail-on-violation

Filtering: Forward Checking: Cross off values that violate a constraint when added to assignment; whenever any variables has no value left, backtrack.

Arc Consistency: An arc X→Y is consistent iff for every x in the tail there is some y in the head could be assigned without violating a constraint.

* Delete from the tail. If Y loses a value, arc X→Y needs to be re-checked.

Ordering: Minimum Remaining Value: the variable with the fewest legal left values in domain

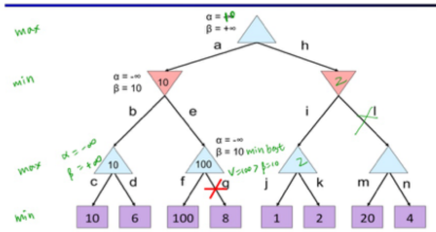
Least Constraining Value: the one rules out the fewest values in the remaining vars.

Structure: Tree structure: Choose a root variable, order variable that parents precede childs

Remove backward: n→2, remove inconsistent + Assign forward: 1→n assign x

Cutset structure: a set of variables s.t. the remaining constraint is a tree.

Alpha-Beta Example 2



Adversarial Search

Resource Limit: ① Depth-limited ② Limiting branching factor

Evaluation Function: $Eval(s) = w_1 f_1(s) + w_2 f_2(s) + \cdots$

Minimax

```
def value(state):
    if the state is a terminal state: return the state's utility
    if the next agent is MAX: return max-value(state)
    if the next agent is MIN: return min-value(state)

def max-value(state):
    initialize v = -∞
    for each successor of state:
        v = max(v, value(successor))
    return v

def min-value(state):
    initialize v = +∞
    for each successor of state:
        v = min(v, value(successor))
    return v
```

α: MAX's best option on path to root
β: MIN's best option on path to root

Alpha-beta

```
def max-value(state, α, β):
    initialize v = -∞
    for each successor of state:
        v = max(v, value(successor, α, β))
        if v ≥ β return v
        α = max(α, v)
    return v

def min-value(state, α, β):
    initialize v = +∞
    for each successor of state:
        v = min(v, value(successor, α, β))
        if v ≤ α return v
        β = min(β, v)
    return v
```

Expectimax: no pruning!

```
def value(state):
    if the state is a terminal state: return the state's utility
    if the next agent is MAX: return max-value(state)
    if the next agent is EXP: return exp-value(state)

def max-value(state):
    initialize v = -∞
    for each successor of state:
        v = max(v, value(successor))
    return v

def exp-value(state):
    initialize v = 0
    for each successor of state:
        p = probability(successor)
        v += p * value(successor)
    return v
```

Propositional Logic

Syntax & semantic: conjunctive: ∧; disjunctive: ∨. Valid / satisfiable / unsatisfiable

Entailment: $\alpha \models \beta$ $model(\alpha) \subseteq model(\beta)$

Proof: $\alpha \vdash \beta$ { model checking ✗, rule of inference }

Sound: $\alpha \vdash \beta \Rightarrow \alpha \models \beta$

Complete: $\alpha \models \beta \Rightarrow \alpha \vdash \beta$

$(\alpha \wedge \beta) \equiv (\beta \wedge \alpha)$ commutativity of ∧
$(\alpha \vee \beta) \equiv (\beta \vee \alpha)$ commutativity of ∨
$((\alpha \wedge \beta) \wedge \gamma) \equiv (\alpha \wedge (\beta \wedge \gamma))$ associativity of ∧
$((\alpha \vee \beta) \vee \gamma) \equiv (\alpha \vee (\beta \vee \gamma))$ associativity of ∨
$\neg(\neg\alpha) \equiv \alpha$ double-negation elimination
$(\alpha \Rightarrow \beta) \equiv (\neg\beta \Rightarrow \neg\alpha)$ contraposition
$(\alpha \Rightarrow \beta) \equiv (\neg\alpha \vee \beta)$ implication elimination
$(\alpha \Leftrightarrow \beta) \equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha))$ biconditional elimination
$\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta)$ de Morgan
$\neg(\alpha \vee \beta) \equiv (\neg\alpha \wedge \neg\beta)$ de Morgan
$(\alpha \wedge (\beta \vee \gamma)) \equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma))$ distributivity of ∧ over ∨
$(\alpha \vee (\beta \wedge \gamma)) \equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma))$ distributivity of ∨ over ∧

$B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$ Convert to CNF

1. Eliminate ⇔, replacing α ⇔ β with (α ⇒ β)∧(β ⇒ α).
   $(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$
2. Eliminate ⇒, replacing α ⇒ β with ¬α∨ β.
   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg(P_{1,2} \vee P_{2,1}) \vee B_{1,1})$
3. Move ¬ inwards using de Morgan's rules and double-negation:
   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge ((\neg P_{1,2} \wedge \neg P_{2,1}) \vee B_{1,1})$
4. Apply distributivity law (∧ over ∨) and flatten:
   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1})$

Prove KB ⊨ α → Show KB∧¬α is unsatisfiable

Horn Logic: 1° $P_1 \wedge P_2 \wedge \cdots \wedge P_n \Rightarrow Q$, 2° $\neg P_1 \vee \neg P_2 \vee \cdots \vee \neg P_n \vee Q$

⇒ Forward chain: Add new clauses into KB until Q is added → data-driving

Backward chain: goal-driving # avoid loop (check if new subgoal is on the goal stack) # avoid repeated work (has already be proved or fail)

First-Order Logic

Syntax: Logical symbols: Connectives (¬, ⇒, ∧, ∨, ⇔) Quantifiers (∀, ∃) Variable (x, y) Equality (=)

Non-logical symbols: Constants (King, 2) Predicates (Brother, >) Function (Sqrt, LeftLegOf)

Term: constant / variable / function(term1, term2, ...)

Atomatic sentence: predicate(term1, term2, ...) / term1 = term2.

Complex sentence: automatic sentences using connectives (¬S, S1⇒S2, ¬(l1, l2) ∨ S(l1, l2)) Sibling(A,B) ⇒ Sibling(B,A)

Semantic: model = objects + relations + interpretation (specify symbols)

∀x At(x, STU) ⇒ Smart(x) ∃x At(x, STU) ∧ Smart(x)

∀x∀y same as ∀y∀x, ∃x∀y not ∀y∃x. ∀x P(x) ≡ ¬∃x ¬P(x)

∃x∃y same as ∃y∃x. ∃x P(x) ≡ ¬∀x ¬P(x)

In FOL, every var must be bound. ∀x P(x, y) is not valid.

Inference. I) Propositionalization (UI / EI)

II) Unification: substitude, MGU = {y/John, x/z}

III) FC/BC (Horn Logic) ← universally quantified.

IV) Resolution

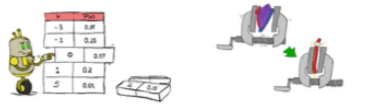
1. Eliminate biconditionals and implications
   ∀x [¬∀y ¬Animal(y) ∨ Loves(x,y)] ∨ [∃y Loves(y,x)]
2. Move ¬ inwards: ¬∀x p ≡ ∃x ¬p, ¬∃x p ≡ ∀x ¬p
   ∀x [∃y ¬(¬Animal(y) ∨ Loves(x,y))] ∨ [∃y Loves(y,x)]
   ∀x [∃y ¬¬Animal(y) ∧ ¬Loves(x,y)] ∨ [∃y Loves(y,x)]
   ∀x [∃y Animal(y) ∧ ¬Loves(x,y)] ∨ [∃y Loves(y,x)]
3. Standardize variables: each quantifier should use different variable
   ∀x [∃y Animal(y) ∧ ¬Loves(x,y)] ∨ [∃z Loves(z,x)]
4. Skolemize: a more general form of existential instantiation. Each existential variable is replaced by a Skolem function of the enclosing universally quantified variables:
   ∀x [Animal(F(x)) ∧ ¬Loves(x,F(x))] ∨ Loves(G(x),x)
5. Drop universal quantifiers:
   [Animal(F(x)) ∧ ¬Loves(x,F(x))] ∨ Loves(G(x),x)
6. Distribute ∨ over ∧ :
   [Animal(F(x)) ∨ Loves(G(x),x)] ∧ [¬Loves(x,F(x)) ∨ Loves(G(x),x)]

Exact Inference

Enumeration: exponential

- General case:
  - Evidence variables: $E_1 \ldots E_k = e_1 \ldots e_k$
  - Query variable: Q
  - Hidden variables: $H_1 \ldots H_r$
  - } $X_1, X_2, \ldots X_n$ All variables
- We want: $P(Q|e_1 \ldots e_k)$

- Step 1: Select the entries consistent with the evidence
- Step 2: Sum out H to get joint of Query and evidence
- Step 3: Normalize



$$\times \frac{1}{Z}$$

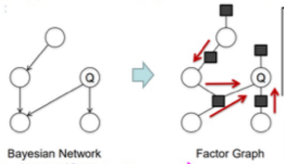
$$P(Q, e_1 \ldots e_k) = \sum_{h_1 \ldots h_r} P(Q, h_1 \ldots h_r, e_1 \ldots e_k)$$
$X_1, X_2, \ldots X_n$

$$Z = \sum_q P(Q, e_1 \cdots e_k)$$
$$P(Q|e_1 \cdots e_k) = \frac{1}{Z} P(Q, e_1 \cdots e_k)$$

Variable Elimination: exponential, often better

$\sum_{L} P(L|t) P(r) P(t|r) \Rightarrow \sum_{L} P(L|t) \sum_{r} P(r) P(t|r)$

Polytree: Directed graph without undirected cycles ⇒ linear complexity



Bayesian Network   Factor Graph

- Input: evidence $e_1, \ldots, e_k$
- For i=1, 2, …, n
  - Sample X_i from $P(X_i \mid parents(X_i))$
  - If x_i not consistent with evidence
    - Reject: Return, and no sample is generated in this cycle
- Return $(x_1, x_2, \ldots, x_n)$

reject

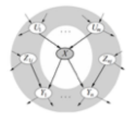
- For i=1, 2, …, n (in topological order)
  - Sample X_i from $P(X_i \mid parents(X_i))$
- Return $(x_1, x_2, \ldots, x_n)$

prior

- Input: evidence $e_1, \ldots, e_k$
- $w = 1.0$
- for i=1, 2, …, n
  - if X_i is an evidence variable
    - x_i = observed value$_i$ for X_i
    - Set $w = w * P(x_i \mid Parents(X_i))$
  - else
    - Sample x_i from $P(X_i \mid Parents(X_i))$
- return $(x_1, x_2, \ldots, x_n)$, w

likelihood

Bayesian Network

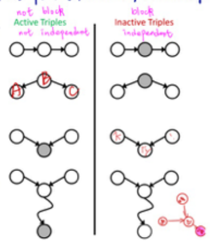


Syntax: DAG + CPTs

Semantic

Markov blanket: independent of all other vars given its parents, children, children's parents

- Question: X, Y, Z are non-intersecting subsets of nodes. Are X and Y conditionally independent given Z?
- A triple is active in the following three cases
  - Causal chain A → B → C where B is unobserved (either direction)
  - Common cause A ← B → C where B is unobserved
  - Common effect (aka v-structure) A → B ← C where B or one of its descendents is observed
- A path is active if each triple along the path is active
- A path is blocked if it contains a single inactive triple
- If all paths from X to Y are blocked, then X is said to be "d-separated" from Y by Z (undirected)
- If d-separated, then X and Y are conditionally independent given Z



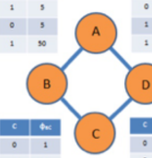
Markov Networks: Undirected graph + potentials

A	B	C	D	$\phi_{AB}\phi_{BC}\phi_{CD}\phi_{AD}$
0	0	0	0	250
0	0	0	1	37500
0	0	1	0	50000
0	0	1	1	625000
0	1	0	0	1125
0	1	0	1	168750
0	1	1	0	50000
0	1	1	1	625000
1	0	0	0	250
1	0	0	1	375
1	0	1	0	50000
1	0	1	1	6250
1	1	0	0	112500
1	1	0	1	168750
1	1	1	0	5000000
1	1	1	1	625000

Z = 7520750

A	B	ϕ_{AB}
0	0	50
0	1	5
1	0	5
1	1	50

A	D	ϕ_{AD}
0	0	5
0	1	50
1	0	50
1	1	5

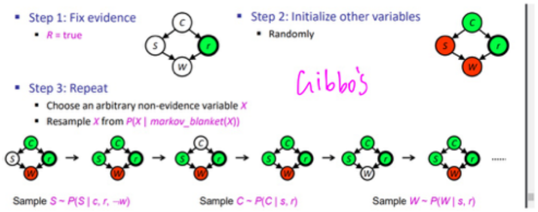


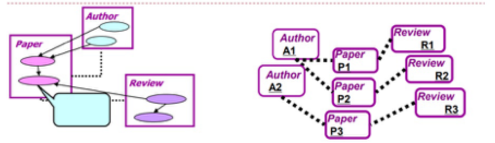
B	C	ϕ_{BC}
0	0	1
0	1	5
1	0	45
1	1	50

C	D	ϕ_{CD}
0	0	1
0	1	15
1	0	40
1	1	50

Gibbs

- Step 1: Fix evidence
  - R = true
- Step 2: Initialize other variables
  - Randomly
- Step 3: Repeat
  - Choose an arbitrary non-evidence variable X
  - Resample X from P(X | markov_blanket(X))





PRM (S, Θ) + relational skeleton (σ) = probability distribution over instantiations of attributes I:

$$P(I \mid \sigma, S, \Theta) = \prod_{x \in \sigma} \prod_{x.A} P(x.A \mid parents_{S,\sigma}(x.A))$$

Objects Attributes