State

Assume: Agent position: 120; Food Count: 30
Ghost position: 12; Agent Face: 4
=> State Num: $120*2^{30}*2^{12}*4$
State space: $n * 2^k$

Search

fringe: 保存即将拓展的节点
complete: 能够找到一个解
optimal: 能够找到最优解

DFS

fringe: LIFO stack
Time: $O(b^S)$ Space: $O(bs)$
complete: complete when No Cycle
Optimal: No

BFS

fringe: FIFO queue
Time: $O(b^m)$ Space: $O(bm)$
complete: complete
Optimal: Yes when all cost is 1

Iterative Deepening

BFS+DFS
1. Run DFS with depth limit 1, if not find, ...
2. Run DFS with depth limit 2, if not find, ...
每一层复杂度指数上升, 因此上一层结果不需要传递给下一层

UCS

fringe: priority queue
cost sensitive BFS
Time: $O(b^{C^*/\epsilon})$ Space: $O(b^{C^*/\epsilon})$
complete: finite cost and all positive
Optimal: Yes

A* Search

combine greedy and UCS
f(n)=g(n)+h(n)
Admissible: optimal in tree search
Consistency: optimal in graph search
reason: 任意路径上cost不下降
Consistency => Admissible
e.g. for consistent: Euclidean/Manhattan

CSP

Standard Search Fomulation: 朴素搜索
给一个变量赋值, 全部结束后判断是否有冲突, 如果有则重新赋值

Backtracking

DFS + variable order + fail-on-violation
只选择与约束不冲突的赋值, 如果全部冲突, 那么backtrack回溯

Filter: Constraints Propagation

1. 初始化. 将所有的Arc都放入一个queue
2. 反复移除Arc: X_i->X_j, 强制要求Arc是consistency的. 即对于每一个 $v \in D(X_i)$ 都有 $w \in D(X_j)$ 能够让(v,w)满足约束
3. 如果v没有任何的w能够使之满足约束, 那么需要从domain中删除
4. 如果删除了任意值, 那么需要将所有的 X_k->X_i 重新放入队列中
5. 重复直到队列为空或者某个domain为空集

时间复杂度: $O(ed^3)=O(n^2d^3)$

```
function AC-3(csp) returns the CSP, possibly with reduced domains
  inputs: csp, a binary CSP with variables {X1, X2, ..., Xn}
  local variables: queue, a queue of arcs, initially all the arcs in csp

  while queue is not empty do
    (Xi, Xj) ← REMOVE-FIRST(queue)
    if REMOVE-INCONSISTENT-VALUES(Xi, Xj) then
      for each Xk in NEIGHBORS[Xi] do
        add (Xk, Xi) to queue
```

```
function REMOVE-INCONSISTENT-VALUES(Xi, Xj) returns true iff succeeds
  removed ← false
  for each x in DOMAIN[Xi] do
    if no value y in DOMAIN[Xj] allows (x,y) to satisfy the constraint Xi ↔ Xj
      then delete x from DOMAIN[Xi]; removed ← true
  return removed
```

K-Consistency:
在remove backward之后的状态.
保证k-1的consistent的赋值一定能拓展到第k个变量上, 而不违反任何约束

互相容Arc Consistency是2-consistency

Ordering: Most Remaining Value MRV

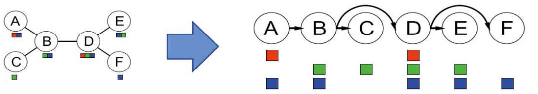
下一个变量选择domain中剩余最多的一个
这个顺序无法提前得知, 因为与已赋值的变量的值有关

Ordering: Least Constraints Value LCV

下一个变量选择约束最少的一个

Structure: Tree-Structured CSP

$O(d^n)$->$O(nd^2)$



1. 将图展平(任意顺序), 然后用有向箭头连接成为一颗树
即, 令无向图线性化
2. Remove Backward
要求所有的Arc: Pa(x)->x是consistent的
3. Assign Forward
在可选的domain中选择一个值赋值
- 在Remove Backward之后, 从Root到Leaf都是arc consistency的
- 如果Root到leaf都是consistency的, 那么Forward Assign不会Backtracking

Cutset Conditioning

找到割集, 为cutset分配变量, 只留下树状CSP
时间复杂度: $O(d^c(n-c)d^2)$

Local Search

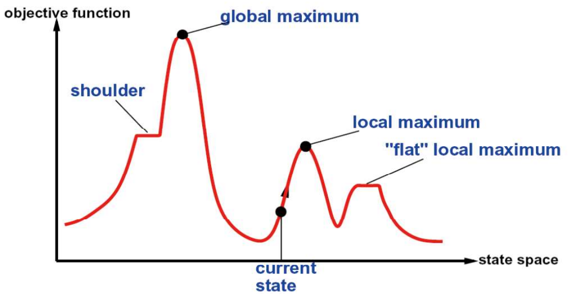
State: 随机对变量完整赋值, Successor: 找到违反约束最多的变量重新赋值

Performance: R 接近 critical ratio时表现较差

$$R = \frac{\text{number of constraints}}{\text{number of variables}}$$

 # R很大和很小都会减少算法运行的时间。R很大时, 变量很少, 会减少运行时间。R很小的时候, 约束稀疏, 可行解范围很大, 运行时间也很短。

Hill Climb



Adversarial Search

Zero Sum

Agents have opposite **utilities** (values on outcomes)

Adversarial, **pure competition**.

Minimax values

Non-Terminal State:

Agent's Control: $V(s) = \max_{s' \in \text{successors}(s)} V(s')$

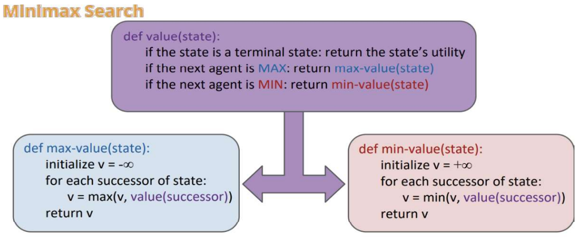
Opponent's Control: $V(s') = \min_{s \in \text{successors}(s')} V(s)$

Terminal State: **# 终端状态, 定值和固定的游戏性质**

Adversial Search(Minimax)

1. **Deterministic, zero-sum**
2. Players alternate turns.
3. One player max result.
4. One player min result.

Minimax Search



Efficiency

Like (exhaustive) DFS

- **Time**: $O(b^m)$
- **Space**: $O(bm)$

Resource Limits

Solution: Depth-limited search

- Replace terminal utilities with an evaluation function for non-terminal positions.

Evaluation Functions

Ideal: Returns the actual minimax value of the position.

Alpha-beta Pruning

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

```
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
```

Max层只更新alpha, min层只更新beta
当alpha >= beta的时候剪枝

Logic

Syntax: 是否是一个合法的语句
Semantic: 这个model是否能让语句是true/false

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

Valid: 是否一定是true
satisfiable: 是否有一个model使之为true
unsatisfiable: 没有任何一个model使之为true

Entail: 如果a中的所有model都是true, 那么b中的也是true
(a包含于b)
Proof: a demonstration of entailment from a to b

soundness 健全: everything can be proved is in fact entailed
completeness 完整: everyuthing that is entailed can be proved
sound意味着所有可证明的都是对的, 即无法证明错误存在
complete意味着所有正确都可以被证明

CNF: 只存在 与或非 的逻辑表达式
应该被叫做 conjunction of disjunction of literals
也叫做Clause

Resolution Rule: an inference rule in PL

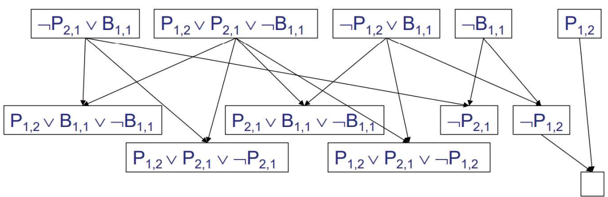
Examples:

$$\frac{P_{1,3} \lor P_{2,2}, \quad P_{2,3} \lor \neg P_{2,2}}{P_{1,3} \lor P_{2,3}} \qquad \frac{P_1, \neg P_1}{\{\}}$$

如果两个clause之间有相互无法证明的内容(如, P_1, $\neg P_1$)那么可以消除掉.
Inference Rule的本质就是两个clause都为true的时候能够推导出其他的一定为true的clause
e.g. 证明: $KB \models \alpha$

1. Convert $KB \land \neg\alpha$ to CNF
2. Repeatedly apply the resolution rule to add new clauses, until one of the two things happens
   a) Two clauses resolve to yield the empty clause, in which case *KB* entails α
   b) There is no new clause that can be added, in which case *KB* does not entail α

$KB = (B_{1,1} \Leftrightarrow (P_{1,2} \lor P_{2,1})) \land \neg B_{1,1}$
$\alpha = \neg P_{1,2}$



– A Horn clause has the form:
$P1 \land P2 \land P3 \ldots \land Pn \Rightarrow Q$
or alternatively
$\neg P1 \lor \neg P2 \lor \neg P3 \ldots \lor \neg Pn \lor Q$

Modus Ponens

$$\frac{\overbrace{\alpha 1, \ldots, \alpha n, \quad \alpha 1 \land \ldots \land \alpha n}^{\text{premises}} \Rightarrow \beta}{\beta} \quad \text{conclusion}$$

Forward Chaining and Backward Chaining run linear time, linear space
Forward Chaining是健全的且完整的, 但是Backward chaining是健全的但是不完整的.
Horn Logic和Inference Rule都是sound & complete
Modus Ponens只对Horn Logic是sound & complete
一个complete的搜索算法可以用于produce complete inference算法
Forward chaining是data-driven, 不断地向knowledge base中添加推导出的逻辑, 直到找到我们想要的
Backward chaining是goal-driven, 我们只证明需要用的逻辑, 然后不断向前推导, 直到所有的subgoal被证明
Backward chaining:
- Avoid loop: 检查subgoal是否已经在需要证明的stack中
- Avoid repeat work: 检查subgoal是否已经被证明(或被证否)

FOL:
- Objects: people, houses, numbers, colors, baseball games, wars, ...
- Relations: red, round, prime, bigger than, part of, comes between, ...
- Functions: father of, best friend of, one more than, ...

Atomic Sentences:
predicate(term1, terms2, ...) or term1=term2

Logical symbols
- Connectives ¬, ∧, ∨, ⇒, ⇔
- Quantifiers ∀, ∃
- Variables x, y, a, b, ...
- Equality =

Non-logical symbols (ontology)
- Constants KingArthur, 2, ShanghaiTech, ...
- Predicates Brother, >, ...
- Functions Sqrt, LeftLegOf, ...

一般使用∀和⇒配合,∃和∧配合

∀x Likes(x,IceCream) ≡ ¬∃x ¬Likes(x,IceCream)
∃x Likes(x,Broccoli) ≡ ¬∀x ¬Likes(x,Broccoli)

Universal Inference(UI):

$$\frac{\forall v\ \alpha}{\text{Subst}(\{v/g\}, \alpha)} \longleftarrow \text{Substitute } v \text{ with } g \text{ in } \alpha$$

Existential Inference(EI):

$$\frac{\exists v\ \alpha}{\text{Subst}(\{v/k\}, \alpha)}$$

其中存在的变量可以用一个函数代替,
称作Skolem constant, 如: C_1

∃x Crown(x) ∧ OnHead(x,John) yields:
Crown(C_1) ∧ OnHead(C_1,John)

Propositional Inference:
如果alpha被FOL KB蕴含, 那么可以由知识库中有限大小的子集蕴含
但是如果alpha并未被KB蕴含, 那么会陷入无限循环

Unification:
将一个变量替换成一个literal常量.
如: King(x)替换成King(John)

p	q	θ
Knows(John,x)	Knows(John,Jane)	{x/Jane}
Knows(John,x)	Knows(y,OJ)	{x/OJ,y/John}
Knows(John,x)	Knows(y,Mother(y))	{y/John,x/Mother(John)}
Knows(John,x)	Knows(x,OJ)	{fail}

在做unify之前应该先standardize, 将两个语句的相同名字的变量替换掉, 因为可能表示两个完全不同的内容
MGU: 在做替换的时候, 保证最泛化的替换
MGU可能不止一个
θ = {y/John, x/z } or θ = {y/John, x/John, z/John}

Horn Logic Inference

$$\frac{p_1', p_2', \dots, p_n', (p_1 \wedge p_2 \wedge \dots \wedge p_n \Rightarrow q)}{q\theta} \quad \text{where } p_i'\theta = p_i\theta \text{ for all } i$$

FOL Forward chaining properties:
1. Sound and complete for FOL Horn clauses
2. FC terminates for first-order Horn clauses with no functions in finite number of iterations
3. In general, FC may not terminate if α is not entailed

Backward Chaining:
1. Depth-first recursive proof search: space is linear in size of proof
2. Avoid infinite loops
3. Avoid repeat works

Conversion to CNF:
1. Eliminate biconditionals and implications
2. move¬inwards: ¬∀x p ≡ ∃x ¬p, ¬∃x p ≡ ∀x ¬p
3. standardize the variables: each quantifier should use a different variable
4. Skolemize: each existential variable should be replaced by a Skolem function of enclosing unversally quantified variable.
5. Distribute disjunction over conjunction
6. Drop unversally quantifier

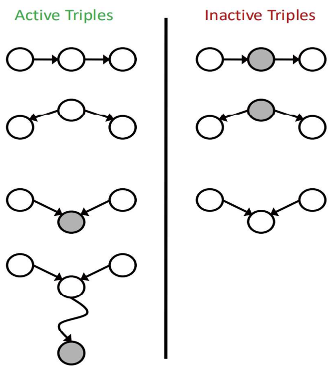
Bayes Network

联合概率密度分布: Time: O(d^n) Space: O(d^n)
有向无环图.
强假设: 每一个点只与自己的父节点相关
CPT: Conditional Probability Table
对于某一个子节点:
- 假设父节点的domain为di
- 假设该节点的domain为d
- 每一行之和是1
- 那么该节点的复杂度(参数量)是(d-1)∏ di
- d-1的原因是行之和为1

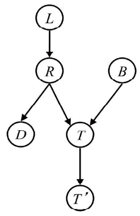
对于一个n节点, 最大domain为d, 最大父节点数量为k的Bayes Net, 空间复杂度为O(nd^{k+1})
So for any i, we have: $P(X_i \mid X_1,\dots,X_{i-1}) = P(X_i \mid Parents(X_i))$

Markov Blanket:
一个节点的父节点, 子节点, 子节点的父节点组成该节点的Markov Blanket
给定Markov Blanket的情况下, 节点与其他节点条件无关

D-separation:
inactive 的path都是独立的



$L \perp\!\!\!\perp T' \mid T$	Yes
$L \perp\!\!\!\perp B$	Yes
$L \perp\!\!\!\perp B \mid T$	
$L \perp\!\!\!\perp B \mid T'$	
$L \perp\!\!\!\perp B \mid T, R$	Yes



Bayes Net causal:
when Bayes Network reflect the true causal patterns:
- often simpler
- often easier to access probability
- often more robust. e.g. change the frequency of one node does not affect rest of model

BNs need not actually be causal
- sometimes no causal net exists over the domain (especially when variables are missing)
- End up with arrows reflect correlation, not causal

Markov Network

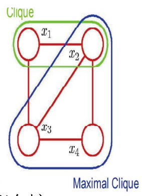
Markov Network = undirect graph + potential function

$$p(\mathbf{x}) = \frac{1}{Z} \prod_C \psi_C(\mathbf{x}_C)$$

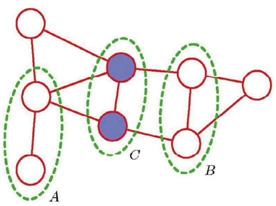
where $\psi_C(\mathbf{x}_C)$ is the potential over clique C and

$$Z = \sum_{\mathbf{x}} \prod_C \psi_C(\mathbf{x}_C)$$

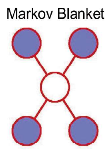
is the normalization coefficient (aka. partition function).



Clique: 一个完全图, 所有点都相互连接
Max Clique: 包含最多点的Clique
protential function: 非负的值



$A \perp\!\!\!\perp B \mid C$



BN to MN:
1. 将所有有向边转换成无向边
2. 将BN中子节点的共同父节点之间连线(moralization)
3. 将CPT转换成protential function

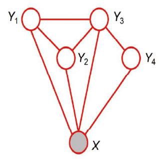
CRF
An extension of MN (aka. Markov random field) where everything is conditioned on an input

$$P(y|x) = \frac{1}{Z(x)} \prod_C \psi_C(y_C, x)$$

where $\psi_C(y_C, x)$ is the potential over clique C and

$$Z(x) = \sum_y \prod_C \psi_C(y_C, x)$$

is the normalization coefficient.



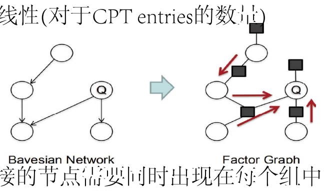
Bayes Network Inference

Variable Elimination(VE):
将求和符尽可能向里面传入
factors: 对于求和过程中, 有可能会出现P(a|B,E)的情况, 有两个变量, 于是可以保留其中一个变量, 叫做factor
消除顺序: 将联合概率按照链式法则拆结成条件概率的求和, 然后消除出现次数最少的一个变量, 通过求和去消除.

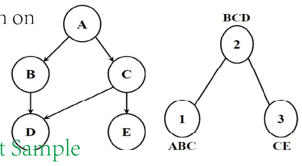
e.g.
Initial factors: $P(+y_2|X_2), P(Y_1|X_1), P(X_1), P(X_2|X_1, Y_1)$
choose to eliminate hidden r.v. Y_1, $P(X_2|X_1) = \sum_{y_1} P(y_1|X_1)P(X_2|X_1, y_1)$
resulting factors: $P(+y_2|X_2), P(X_2|X_1), P(X_1)$
choose to eliminate hidden r.v. X_1, $P(X_2) = \sum_{x_1} P(X_2|x_1)P(x_1)$
resulting factors: $P(+y_2|X_2), P(X_2)$
choose to eliminate hidden r.v. X_2, $P(+y_2) = \sum_{x_2} P(+y_2|x_2)P(x_2)$

并不一定存在一个得到最少factor的顺序
there not always exist an ordering that only results in small factors

Poly-Tree: 一个有向图, 但是不存在无向环
对于Poly-Treee BN, VE可以是线性(对于CPT entries的数量)复杂度, 在以下顺序:
- convert to factor graph
- Take one as root
- eliminate from leaves to root



Junction Tree
分成多个组, 每个组之间相连接的节点需要同时出现在每个组中
- Run a sum-product-like algorithm on the junction tree.
- Intractable on graphs with large cluster nodes



Prior Sample
直接采样
根据频率获取概率
低概率事件不容易采样

Reject Sample
1. 直接采样
2. 遇到变量不是我们想要的值就直接结束
3. 根据频率获取概率
低概率事件不容易采样

Likelihood Sample
1. 固定已知的变量
2. 按照随机数赋值
3. 如果遇到固定的变量, 那么按照条件概率乘到weight中
4. 按照weight来normalize, 然后计算对应的probability

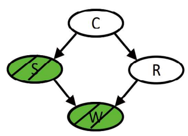
Important Sample:
如果原始的概率P(x)采样比较困难, 那么考虑使用Q(x)来采样, 那么weight应该变成了P(x)/Q(x)
Q(x)的选取对算法的影响很大, 最好的Q(x)应该有Q(x)~|f(x)|P(x)

- Sampling distribution (z is sampled and e is fixed evidence)

$$S_{WS}(\mathbf{z}, \mathbf{e}) = \prod_{i=1}^{l} P(z_i|\text{Parents}(Z_i))$$

- Now, samples have weights

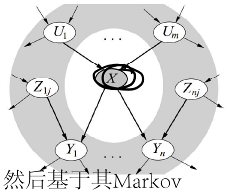
$$w(\mathbf{z}, \mathbf{e}) = \prod_{i=1}^{m} P(e_i|\text{Parents}(E_i))$$



- Together, weighted sampling distribution is consistent

$$S_{WS}(z, e) \cdot w(z, e) = \prod_{i=1}^{l} P(z_i|\text{Parents}(z_i)) \prod_{i=1}^{m} P(e_i|\text{Parents}(e_i))$$
$$= P(\mathbf{z}, \mathbf{e})$$

Gibbs Sample
$X_i' \sim P(X_i \mid x_1,\dots,x_{i-1},x_{i+1},\dots,x_n)$
In a Bayes net
$P(X_i \mid x_1,\dots,x_{i-1},x_{i+1},\dots,x_n)$
$= P(X_i \mid markov\_blanket(X_i))$
$= \alpha\ P(X_i \mid u_1,\dots,u_m)\ \prod_j P(y_j \mid parents(Y_j))$



1. 完全随机初始化所有的变量
2. 随机指定一个变量, 移除该变量的赋值, 然后基于其Markov Blanket进行采样
3. 重复上述步骤多次之后能够得到一个近似与真实概率下的分布
4. 上述过程称为warmup.
然后开始采样, 并根据采样进行计算概率