Constraint satisfaction problems

Factor graph - (aka Markov random field) a set of variables $X = X_1, \dots, X_n$ where $X_i \in \text{Domain}_i$ and factors $f_1, \dots, f_m$, with each $f_j(X) \geq 0$. Each factor is implemented as checking a solution rather than computing the solution.

Domain - possible values to be assigned to a variable.

Scope of a factor f_j - the set of variables f_j depends on. **Arity -** the size of this set. "Unary factors" (arity 1); "Binary factors" (arity 2). "Constraints" (factors that return 0 or 1).

Assignment weight - each assignment $x = (x_1, \dots, x_n)$ yields a $\text{Weight}(x)$ defined as being the product of all factors f_j applied to that assignment.

$\boxed{\text{Weight}(x) = \Pi_{j=q}^{m} f_j(x)}$ (x in its entirety is passed in to each f_j for simplicity of this notation, though in reality only a subset of x would be needed for f_j)

CSP - a factor graph where all factors are binary.

$\boxed{\text{For } j = 1, \cdots, \; f_j(x) \in \{0, 1\}}$ (the constraint j with assignment x is said to be satisfied iff $f_j(x) = 1$.)

Consistent assignment x of a CSP - iff $\text{Weight}(x) = 1$ (i.e., all constrains are satisfied.)

Dependent factors $D(x, X_i)$ - a set of factors depending on X_i but not on unassigned variables.

Backtracking search - find maximum weight assignment of a factor graph.

Backtrack$(x, w, \textbf{Domains})$

1. choose an unassigned **variable** X_i (MCV)
2. order **values** of X_i's Domain (LCV)
3. for each value v in the order:
   (a) $\delta \leftarrow \Pi_{f_j \in D(x, X_i)} f_j(x \cup \{X_i : v\})$
   (b) if $\delta = 0$: continue
   (c) $\text{Domains}' \leftarrow$ Domains via **lookahead** (forward checking)
   (d) if $\text{Domains}'_i$ is empty: continue
   (e) **Backtrack**$(x \cup \{X_i : v\}, w\delta, \textbf{Domains}')$

- Strategy: extends partial assignments
- Optimality: exact
- Time: exponential

Forward checking - one-step lookahead heuristic that preemptively removes inconsistent values from the domains of neighboring variables.

- After assigning a variable X_i, it eliminates inconsistent values from the domains of all its neighbors.
- If any of these domains become empty, stop the local backtracking search.
- if we unassign a variable X_i, have to restore the domain of its neighbors.

Most constrained variable - selects the next unassigned variable that has the fewest consistent values: fail early, prune early.

Least constrained value - assigns the next value that yields the highest number of consistent values of neighboring variables: prefers the value that is most likely to work.

Arc consistency of variable X_i - w.r.t. X_j is enforced when for each $x_i \in \text{Domain}_i$, there exists $x_j \in \text{Domain}_j$ such that any factos between X_i and X_j is non-zero.

AC-3 - a multi-step lookahead heuristic that applies forward checking to all relevant variables. After a given assignment, it performs forward checking and then successively enforces arc consistency w.r.t. the neighbors of variables for which the domain change during the process. AC-3 only looks locally at the graph for nothing blatantly wrong; it can't detect when there are no consistent assignments.

Beam search - extends partial assignments of n variables of branching factor $b = |\text{Domain}|$ by exploring the K top paths at each step. The beam size $1 \geq K \geq b^n$ controls the tradeoff between efficiency and accuracy. Runtime is Linear to n: $O(n \underbrace{Kb \log(Kb)}_{\text{sorting top K}})$. $K = 1$: greedy search ($O(nb)$ time); $K \to +\infty$: BFS ($O(b^n)$ time).

- Strategy: extends partial assignments
- Optimality: approximate
- Time: linear

Local search (iterated conditional modes) - modifies the assignment of a factor graph one variable at a time until convergence. AT step i, assign to X_i the value v that maximizes the product of all factors connected to that variable. ICM may get stuck in local optima; adding randomness may help.

- Strategy: modify complete assignments
- Optimality: approximate
- Time: linear

Markov Networks

CSPs	Markov networks
variables	random variables
weights	probabilities
max weight assignment	marginal probabilities

Capture the uncertainty over assignment using the language of probability.

Markov Network - a factor graph which defines a joint distribution over random variables $X = (X_1, \dots, X_n)$: $\boxed{\mathbb{P}(X = x) = \frac{\text{Weight}(x)}{Z}}$

$Z = \sum_{x'} \textbf{Weight}(x')$ **-** sum all the possible assignments' weights (normalization constant)

Marginal probability - the probability of when one particular variable X_i is assigned with a particular value v: sum $\mathbb{P}$ when $X_i = v$

$\boxed{\mathbb{P}(X_i = v) = \sum_{x: x_i = v} \mathbb{P}(X = x)}$

Gibbs sampling - Initialize x to a random complete assignment. Loop through $i = 1, \dots, n$ until convergence:

- Set $x_i = v$ with probability $\mathbb{P}(X_i = v | \underbrace{X_{-i}}_{\text{all vars except} X_i} = x_{-i})$
- Increment $\text{count}_i(x_i)$ (how often this assignment is encountered. can just track particular vars we're interested in.)

Estimate $\hat{\mathbb{P}}(X_i = x_i) = \frac{\text{count}_i(x_i)}{\sum_v \text{count}_i(v)}$

ICM	Gibbs sampling
max weight assignment	marginal probabilities
choose best value	sample a value
converges to local optimum	marginals converge to correct answer

Bayesian Networks

Explaining away - suppose two causes positively influence an effect. Conditioned on the effect, further conditioning on one causes reduces the probability of the other cause.

Bayesian network - a durected acyclic graph that specifies a joint distribution over random variables $X = (X_1, \dots, X_n)$ as a product of local conditional distributions, one for each node:

$\boxed{\mathbb{P}(X_1 = x_1, \dots, X_n = x_n) = \prod_{i=1}^{n} p(x_i | x_{\text{Parents}(i)})}$

Probabilistic program - randomizes variable assignment such that we can write down complex Bayesian networks that generates assignments without having to explicitly specify associated probabilities. Unlike normal classification (e.g., neural nets), Bayesian networks provide a different paradigm where we think about going from output to the input.

Probabilistic inference strategy - to compute the probability $P(Q|E = e)$ of query Q given evidence $E = e$:

1. Remove vars that aren't ancestors of the query Q or the evidence E by marginalization
2. Convert Bayesian network to factor graph
3. Condition on the evidence $E = e$
4. Remove nodes disconnected from the query Q by marginalization
5. Run probabilistic inference algorithm

Filtering question - asks for the distribution of some hidden variable H_i conditioned on only the evidence up until that point. Useful for real-time object tracking as the future can't be seen.

Smoothing question - asks for the distribution of some hidden variable H_i conditioned on on the evidence including the future. Useful when all the data have been collected and we want to retrospectively go and figure out what the hidden state H_i was.

Forward-backward algorithm - computes the exact value of $P(H = h_k | E = e)$ a smoothing query) for any $k \in \{1, \dots, L\}$ in the case of an HHM of size L.

1. for $i \in \{1, \dots, L\}$, compute $F_i(h_i) = \sum_{h_{i-1}} F_{i-1}(h_{i-1}) p(h_i | h_{i-1}) p(e_i | h_i)$
2. for $i \in \{L, \dots, 1\}$, compute $B_i(h_i) = \sum_{h_{i+1}} B_{i+1}(h_{i+1}) p(h_{i+1} | h_i) p(e_{i+1} | h_{i+1})$
3. for $i \in \{1, \dots, L\}$, compute $S_i(h_i) = \frac{F_i(h_i) B_i(h_i)}{\sum_{h_i} F_i(h_i) B_i(h_i)}$