Constraint satisfaction problems

Factor graph - (aka Markov random field) a set of variables $X = X_1, \ldots, X_n$ where $X_i \in \text{Domain}_i$ and factors $f_1, \ldots, f_m$, 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 $x = (x_1, \ldots, x_n)$ yields a 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 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($x, w,$ **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) $\delta \leftarrow \Pi_{f_i \in D(x, X_i)} f_j(x \cup \{X_i : v\})$
   (b) if $\delta = 0$: continue
   (c) Domains$'$ ← Domains via **lookahead** (forward checking)
   (d) if Domains$'_i$ is empty: continue
   (e) **Backtrack**($x \cup \{X_i : v\}, w\delta,$ **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