Lecture 2: Factored discrete spaces

1 Factored discrete problems

- A set of variables $X_1, \ldots, X_n$
- A domain of possible values for each variable, $D_1, \ldots, D_n$
- Evaluation criterion f that maps elements of $D_1 \times \ldots \times D_n$ into $\mathfrak{R}$.

1.1 Example problems

Assigning courses to rooms

Given a set of courses that need scheduling, with expected enrollments, and a set of rooms (with capacities), find a good assignment.

Line-segment matching

Given a set of line-segments found in an image and one or more 2D models of objects in terms of line segments, find the best interpretation of the image.

Sudoku

Given a 9 x 9 Sudoku grid, fill it in following the rules.

2 Constraint satisfaction

- A set of variables $X_1, \ldots, X_n$
- A domain of possible values for each variable, $D_1, \ldots, D_n$ (can extend to infinite domains, e.g., integers)
- A set C of constraints that specify allowable combinations of values

A constraint might look like $\langle (X_1, X_2), \{(A, B), (B, A)\}\rangle$.

A *unary* constraint only involves one variable. A *binary* constraint only involves two variables. A CSP with higher-order constraints can always be converted into one with only binary constraints, by adding variables (that take on, e.g. pairs of values).

When are problems easy? Hard?

2.1 Constraint propagation

Sometimes we can do direct inference to rule out values for variables or even solve the whole problem.

Node consistency

Update the domains of every variable to be consistent with unary constraints.

Arc consistency

X_i is arc-consistent with respect to X_j iff for every value in D_i there is some other value in D_j that satisfies the binary constraint on X_i, X_j.

```
def AC3(csp):
    queue = all binary constraints
    while not empty(queue):
        (Xi, Xj) = queue.pop()
        if revise(csp, Xi, Xj):
            if D_i is empty: then return False
            for each Xk in Xi.neighbors - {Xj} do:
                queue.push((Xk, Xi))
    return true

def revise(csp, Xi, Xj):
    revised = false
    for each xi in Di do:
        if no value xj in Dj allows (xi,xj) to satisfy constraint(Xi, Xj) then:
            Di.delete(xi)
            revised = True
    return revised
```

Run time $O(cd^3)$ where c is the number of binary constraints and d the size of the biggest domain.

Other methods

Look at longer paths and enforce consistency; enforce global constraints (such as that every variable have a different value).

2.2 Backtracking search

In the end, constraint propagation may not solve your problem, and you'll have to search. The combination of search and constraint propagation can be very powerful. One good strategy is to do unary constraints to start with, but that's all.

```
def backtrack(assignment, csp):
    if complete(assignment): return assignment
    v = selectUnassignedVar(assignment, csp)
    for each value in sortDomainValues(var, assignment, csp) do:
```

```
        if consistent(value, assignment):
            newAssignment = assignment.assign(var, value)
            newCSP = inference(copy(csp), var, value)
            if not failed(newCSP):
                result = backtrack(newAssignment, newCSP)
                if result: return result
    return failure
```

Interesting questions (and glib answers):

- What variable to pick next? (most constrained)
- What value to try first? (least constraining)
- What inference should be performed? (forward checking: arc consistency on the variable just assigned)
- Smarter ways to backtrack?

2.3 Decomposition

Other interesting strategies based on the structure of the graph of constraints. Analogous to exact inference methods in Bayesian networks.

3 Assignment trees

When we can write

$$f(x_1, \ldots, x_n) = \sum_{i=1}^{n} f_i(x_i)$$

Or, even,

$$f(x_1, \ldots, x_n) = \sum_{i=1}^{n} f_i(x_1, \ldots, x_i)$$

for some ordering of the variables, then we can build a search tree similar to the one we build for CSP.

Example of edge matching.

Use partial score plus a heuristic cost estimate to order the search. Stop when a complete solution with a lower score than any pending partial solutions (plus heuristic) is found.

4 Belief propagation

Convert SAT problems (CSP with binary domains) into probability distribution over solution space, with uniform dist over satisfying assignments and 0's elsewhere. Estimate θ_i, which is the probability that x_i is positive in an assignment that is a solution.

Factor graph representation

Belief propagation algorithm is exact in trees

Decimation: after BP converges, find the variable whose θ is closest to 0 or 1; fix the value of that variable; generate a new problem with one fewer variable, and repeat.

Not guaranteed to converge.

"Solving Constraint Satisfaction Problems through Belief Propagation-guided Decimation," Montanari, Ricci-Tersenghi, and Semerjian, **arXiv:0709.1667v2**, 2007.

Survey propagation seems to be more reliable on hard problems.

"Survey propagation: an algorithm for satisfiability," Braunstein, Mezard, Zecchina, **arXiv:cs/0212002v4**, 2006.

"A new look at Survey Propagation and its Generalizations," Maneva, Mossel, Wainwright, **arXiv:cs/0409012v3**, 2005.

5 Local methods

Can apply all of our old friends:

- Hill climbing

  Min conflicts heuristic: pick a variable and assign it the value that causes the fewest constraint violations.
- Simulated annealing
- Genetic algorithm...