Constraint Propagation

(Where a better exploitation of the constraints further reduces the need to make decisions)

R&N: Chap. 5 + Chap. 24, p. 881-884

... is the process of determining how the constraints and the possible values of one variable affect the possible values of other variables

It is an important form of "least-commitment" reasoning

Forward checking is only on simple form of constraint propagation

When a pair (X,v) is added to assignment A do:
    For each variable Y not in A do:
        For every constraint C relating Y to variables in A do:
            Remove all values from Y's domain that do not satisfy C

- $n =$ number of variables
- $d =$ size of initial domain
- $s =$ maximum number of constraints involving a given variable ($s \leq n-1$)
- Forward checking takes $O(nds)$ time

Forward Checking in Map Coloring

Empty set: the current assignment

{} does not lead to a solution

Detecting this contradiction requires a more powerful constraint propagation technique
**Constraint Propagation for Binary Constraints**

REMOVE-VALUES(X,Y)
1. removed $\leftarrow$ false
2. For every value $v$ in the domain of $Y$ do
   - If there is no value $u$ in the domain of $X$ such that the constraint on $(X,Y)$ is satisfied then
     a. Remove $v$ from $Y$'s domain
     b. removed $\leftarrow$ true
3. Return removed

**Constraint Propagation for Binary Constraints**

AC3
1. Initialize queue $Q$ with all variables (not yet instantiated)
2. While $Q \neq \emptyset$ do
   a. $X \leftarrow \text{Remove}(Q)$
   b. For every (not yet instantiated) variable $Y$ related to $X$ by a (binary) constraint do
      - If REMOVE-VALUES($X,Y$) then
        i. If $Y$'s domain = $\emptyset$ then exit
        ii. Insert($Y,Q$)

**Edge Labeling**

We consider an image of a scene composed of polyhedral objects such that each vertex is the endpoint of exactly three edges.

An "edge extractor" has accurately extracted all the visible edges in the image. The problem is to label each edge as convex (+), concave (-), or occluding (Æ) such that the complete labeling is physically possible.

The arrow is oriented such that the object is on the right of the occluding edge.
One Possible Edge Labeling

Junction Types

Junction Label Sets

Edge Labeling as a CSP
- A variable is associated with each junction
- The domain of a variable is the label set associated with the junction type
- Constraints: The values assigned to two adjacent junctions must give the same label to the joining edge

AC3 Applied to Edge Labeling

Q = (X₁, X₂, X₃, ...)

AC3 Applied to Edge Labeling

Q = (X₁, ...)
\[ Q = (X_1, \ldots) \]

\[ Q = (X_5, \ldots) \]

\[ Q = (X_3, \ldots) \]
Complexity Analysis of AC3

- \( n \) = number of variables
- \( d \) = size of initial domains
- \( s \) = maximum number of constraints involving a given variable (\( s \leq n - 1 \))
- Each variables is inserted in \( Q \) up to \( d \) times
- REMOVE-VALUES takes \( O(d^2) \) time
- AC3 takes \( O(n \times d \times s \times d^2) = O(n \times s \times d^3) \) time
- Usually more expensive than forward checking

Is AC3 all that we need?

- No !!!
- AC3 can't detect all contradictions among binary constraints
Is AC3 all that we need?

- No!!
- AC3 can’t detect all contradictions among binary constraints

\[ X \neq Y \begin{cases} 1, 2 \end{cases} \]

\[ X \neq Z \begin{cases} 1, 2 \end{cases} \]

\[ Y \neq Z \begin{cases} 1, 2 \end{cases} \]

REMOVE-VALUES(X,Y,Z)

1. removed \( \leftarrow \) false
2. For every value \( w \) in the domain of \( Z \) do
   - If there is no pair \( (u, v) \) of values in the domains of \( X \) and \( Y \) verifying the constraint on \((X,Y)\) such that the constraints on \((X,Z)\) and \((Y,Z)\) are satisfied then
     a. Remove \( w \) from \( Z \)’s domain
     b. removed \( \leftarrow \) true
3. Return removed

Is AC3 all that we need?

- No!!
- AC3 can’t detect all contradictions among binary constraints

\[ X \neq Y \begin{cases} 1, 2 \end{cases} \]

\[ X \neq Z \begin{cases} 1, 2 \end{cases} \]

\[ Y \neq Z \begin{cases} 1, 2 \end{cases} \]

\[ Z \]

\[ (1, 2) \]

Not all constraints are binary

Tradeoff

Generalizing the constraint propagation algorithm increases its time complexity

\rightarrow \text{Tradeoff between time spent in backtracking search and time spent in constraint propagation}

A good tradeoff when all or most constraints are binary is often to combine backtracking with forward checking and/or AC3 (with \text{REMOVE-VALUES} for two variables)

Modified Backtracking Algorithm with AC3

CSP-BACKTRACKING(A, var-domains)

1. If assignment \( A \) is complete then return \( A \)
2. Run AC3 and update var-domains accordingly
3. If a variable has an empty domain then return failure
4. \( X \leftarrow \) select a variable not in \( A \)
5. \( D \leftarrow \) select an ordering on the domain of \( X \)
6. For each value \( v \) in \( D \) do
   a. Add \((X \leftarrow v)\) to \( A \)
   b. var-domains \( \leftarrow \) forward checking(var-domains, \( X \), \( v \), \( A \))
   c. If no variable has an empty domain then
     (i) result \( \leftarrow \) CSP-BACKTRACKING(A, var-domains)
     (ii) If result \( \neq \) failure then return result
   d. Remove \((X \leftarrow v)\) from \( A \)
7. Return failure

A Complete Example: 4-Queens Problem

1) The modified backtracking algorithm starts by calling AC3, which removes no value

2) The backtracking algorithm then selects a variable and a value for this variable. No heuristic helps in this selection. \( X_i \) and the value 1 are arbitrarily selected.
3) The algorithm performs forward checking, which eliminates 2 values in each other variable’s domain.

4) The algorithm calls AC3, which eliminates 3 from the domain of $X_2$, and 2 from the domain of $X_3$, and 4 from the domain of $X_3$.

5) The domain of $X_3$ is empty $\Rightarrow$ backtracking.
4-Queens Problem

6) The algorithm removes 1 from $X_1$’s domain and assign 2 to $X_1$

7) The algorithm performs forward checking

8) The algorithm calls AC3

8) The algorithm calls AC3, which reduces the domains of $X_3$ and $X_4$ to a single value

Exploiting the Structure of CSP

If the constraint graph contains several components, then solve one independent CSP per component

If the constraint graph is a tree, then:

1. Order the variables from the root to the leaves
   $\Rightarrow (X_1, X_2, \ldots, X_n)$

2. For $j = n, n-1, \ldots, 2$ call REMOVE-VALUES($X_j$, $X_i$)
   where $X_i$ is the parent of $X_j$

3. Assign any valid value to $X_1$

4. For $j = 2, \ldots, n$ do
   Assign any value consistent with the value assigned to its parent $X_i$
Exploiting the Structure of CSP

Whenever a variable is assigned a value by the backtracking algorithm, propagate this value and remove the variable from the constraint graph.

If the graph becomes a tree, then proceed as shown in previous slide.