Backjumping

- max-check[i] for V[i]: the deepest variable V[h] that precludes (eliminates) a candidate value for the current variable and if V[h] reinstantiated, a consistent value may be found for V[i].
- if V[h] has no more values, one-step backtrack to V[h-1].
- Backjumping: jumps, then steps back.
- V[h] is not necessarily the reason for conflict.

Conflict-Directed Backjumping

- conf-set[i] for V[i]: records the ***set*** of past variables that failed consistency checks with V[i]. (conf-set[i] initialized to $\{0\}$)

- Upon failure at V[i], CBJ backtracks to V[h], the deepest variable in conf-set[i], and conf-set[i] is carried over to V[h].

Backmarking

Attempts at minimizing redundant consistency checks
V[i]: current variable. V[h]: past variable.

- Situation 1: V[i] $\leftarrow$ x has failed with V[h] $\leftarrow$ y. As long as V[h] $\leftarrow$ y, do not test V[i] $\leftarrow$ x.
- Situation 2: V[i] $\leftarrow$ x has succeeded with V[h] $\leftarrow$ y. As long as V[h] $\leftarrow$ y, do not test V[i] $\leftarrow$ x.

Data structures for BM

- Maximum checking level (mcl): matrix of size n$\times$d.
  mcl[i,k]: the deepest variable V[i] $\leftarrow$ k checked against.
  Compare max-check[i] and mcl[i,k]
- Minimum backup level (mbl): vector of size n.
  mbl[i]: the shallowest variable that has changed value since V[i] was current (marks how far backtrack has gone).

Use

- mcl[i,k]<mbl[i] $\longrightarrow$ do not consider V[i] $\leftarrow$ k.
- mcl[i,k]$\geq$mbl[i] $\longrightarrow$ perform consistency check for V[i] only against variables between mbl[i] and i.

Forward checking: lookahead

When V[i] $\leftarrow$ k is attempted, for future variables j

- reductions[j]:
  sequence of sequences. Each sequence is a set of values removed by a checking against one past variable.
- future-fc[i]:
  set of future variables connected to V[i]
- past-fc[j]: (used with BJ, CBJ)
  set of past variables connected to V[j]

Lookahead strategies

[Haralick & Elliott, AIJ 80]

- Full lookahead $\longrightarrow$ AC
- Partial lookahead
- Forward checking

Strategy:
- BT: systematic, vanilla flavor
- All others: invest in data structures for future returns

Focus:
- BM, FC: reduce consistency checking
- BJ, CBJ: improve backtracking

Forward move:
- BM, FC: cautious
- BT, BJ, CBJ: same forward move

Backward move:
- BT, BM, FC: chronological, same backward move
- BJ, CBJ: intelligent/informed

'They look:'
- BT, BM, BJ, CBJ: backwards
- FC: forward

CSP parameters $\langle n, a, t, d \rangle$

- n is number of variables
- a is maximum domain size
- t is constraint tightness

$$t = \frac{|forbidden\ tuples|}{|all\ tuples|}$$

- d is constraint density

$$d = \frac{e - e_{min}}{e_{max} - e_{min}},$$

where e is the number of constraints,
$e_{min} = (n-1)$, and
$e_{max} = \frac{n(n-1)}{2}$

Alternatively, $\langle n, m, p_1, p_2 \rangle$

- n is number of variables
- m is the number of values per variable (assuming all variables have the same domain size)
- p_1 is the probability of existence of a constraint between two variables
- p_2 is the probability that two values for two variables are in conflict (assuming that a constraint exists between the two variables).

We show that $p_1 \geq d$ for all graphs and
$p_1 \approx d$ for graphs of sizable n.

Graph measures

- width
- induced width
- bandwidth

As an indication of problem difficulty:

- maximum number of constraints per variable
- average number of constraints per variable

Comparison parameters [Haralick & Elliott, AIJ 80]

- # consistency checks for finding all solutions
- # table lookups used in finding all solutions
- # nodes in tree for finding all solutions

- # consistency checks at each level of the tree
- # nodes at each level of the tree
- # table lookups at each level of the tree

+ 3 other parameters..

Given $\mathcal{P} = (\mathcal{V}, \mathcal{D}, \mathcal{C})$ $\begin{cases} \mathcal{V} = \{V_1, V_2, \ldots, V_n\} \\ \mathcal{D} = \{D_{V_1}, D_{V_2}, \ldots, D_{V_n}\} \\ \mathcal{C} \text{ set of constraints} \end{cases}$

Query: can we find one value for each variable such that all constraints are satisfied?

The query could be to determine:

- if the CSP is solvable.
- one, alternatively several, global consistent solution (**exemplification** CSP).
- all global consistent solutions (**enumeration** CSP).
- if a certain tuple is a solution (**decision** CSP).