Solved and Failed CSP's

- C a constraint on variables $y_1, \ldots, y_k$ with domains $D_1, \ldots, D_k$, so $C \subseteq D_1 \times \ldots \times D_k$.
- C is solved if $C = D_1 \times \ldots \times D_k$.
- CSP is solved if
  - all its constraints are solved,
  - no domain of it is empty.
- CSP is failed if
  – it contains the false constraint $\perp$,
  or
  – some of its domains is empty.

Constraint Programming: Basic Framework

- Formulate your problem as a CSP;

```
Solve:
VAR continue: BOOLEAN;
continue:= TRUE;
WHILE continue AND NOT Happy DO
    Preprocess;
        Constraint Propagation;
    IF NOT Happy
    THEN
       IF Atomic
       THEN
           continue:= FALSE
       ELSE
           Split;
           Proceed by Cases
       END
    END
END
```

- *continue* is local to Solve.
- Proceed by Cases leads to a recursive call of Solve for each newly formed CSP.

Preprocess

Bring to desired syntactic form.

Example: Conjunctive normal form

Desired syntactic form: conjunction of clauses, which are disjunctions of literals

$(x \vee y) \wedge (\neg x \vee y \vee z) \wedge (\neg x \vee \neg z)$

Happy

- Found a solution,
- Found all solutions,
- Found a solved form from which one can generate all solutions,
- Determined that no solution exists (inconsistency),
- Found best solution,
- Found all best solutions.
- Reduced all interval domains to sizes $< \varepsilon$.

Atomic

Check

• whether CSP is amenable for splitting, or

• whether search ‘under’ this CSP is still needed.

Split

Split a domain.

- D finite (Enumeration)

$$\frac{x \in D}{x \in \{a\} \mid x \in D - \{a\}}$$

- D finite (Labeling)

$$\frac{x \in \{a_1, \ldots, a_k\}}{x \in \{a_1\} \mid \ldots \mid x \in \{a_k\}}$$

- D interval of reals (Bisection)

$$\frac{x \in [a,b]}{x \in [a, (a+b)/2] \mid x \in [(a+b)/2, b]}$$

Split

- Split a constraint.
  - Disjunctive constraints

    Example:

    Start[task1] + Duration[task1] ≤ Start[task2] ∨
    Start[task2] + Duration[task2] ≤ Start[task1]

    C1 ∨ C2

    C1 | C2

  - Constraints in “compound” form

    Example:

    |p(x)| = a

    p(x) = a | p(x) = − a

Effect of Split

- Each call to Split replaces current CSP P by CSP's P1, . . ., Pn such that the union of P1, . . ., Pn is equivalent to P.

Example Enumeration:

It replaces $\langle C \,;\, DE, x \in D\rangle$ by
$\langle C' \,;\, DE, x \in \{a\}\rangle$ and $\langle C'' \,;\, DE, x \in D - \{a\}\rangle$.

Where C' and C'' are restrictions of the constraints from C to the new domains.

• Split also determines in which operation is to be applied next.

Heuristics

Which

- variable to choose,
- value to choose,
- constraint to split.

Examples:

Select a variable that appears in the largest number of constraints (most constrained variable).

For a domain being an integer interval: select the middle value.

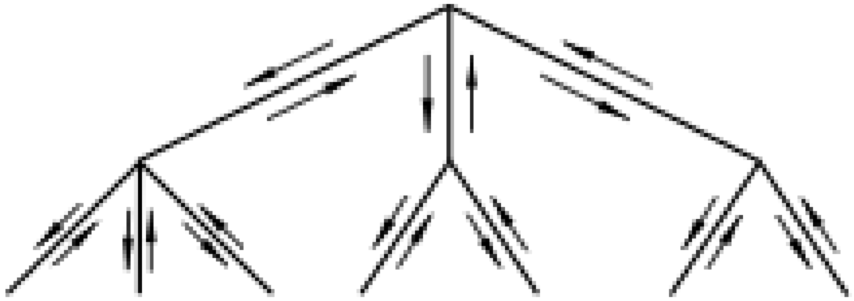
Proceed by Cases

Input: tree of CSP’s (=result of split)
Goal: is to traverse this tree.

Various search techniques.

- Backtracking,
- Branch and bound,
- Can be combined with Constraint Propagation
- Intelligent backtracking

Backtracking

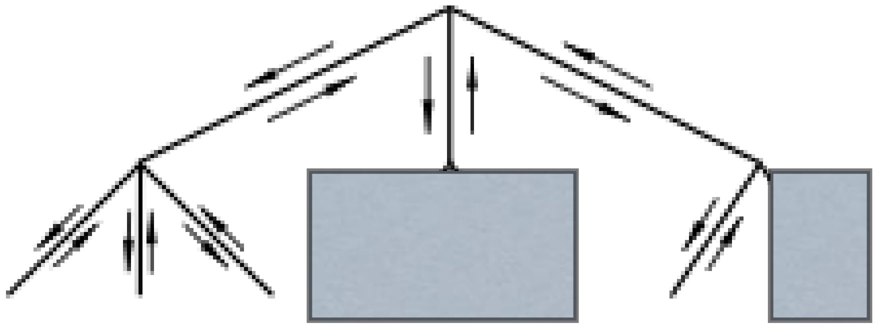


Here

- Nodes generated “on the fly”.
- Nodes are CSP’s.
- Leaves are CSP’s that are solved or failed.

Branch and bound search

- based on backtracking search
- takes into account value of objective function
- aiming at finding the optimal (here maximal) solution.
- during search maintain the currently best value of the objective function
- often used in combination with heuristic function that assigns a value to each considered CSP
- correct use of heuristic funtion h:
  - if the CSP p1 a direct descendent of the CSP p2 in the search tree, then h(p1)<=h(p2)
  - if p1 is “atomic” CSP, then obj(p1) <=h(p1)
- correct use of heuristic function enables to ignore some parts of the tree (h prunes the search tree)



Intelligent backtracking

- faster backtracking
- in case of a failed CSP a jump further back in the tree than just to the parent

Constraint Propagation

- constraint propagation = replace a CSP in a ‘simple’ CSP
- idea: result in smaller search space for Split and Proceed by cases
- “simpler” depends on application
  (often domains and/or constraints become smaller)
  - example: choose a variable x from a constraint C and remove from D_x all the values that can not satisfy the constraint C.
  - example: resolution

Constraint propagation: Reduce a Domain

Linear inequalities on integers.

$\langle x < y \,;\, x \in [50..200],\ y \in [0..100] \rangle$

$\langle x < y \,;\, x \in [50..99],\ y \in [51..100] \rangle$

More generally:

$\langle x < y \,;\, x \in [l_x..h_x],\ y \in [l_y..h_y] \rangle$

$\langle x < y \,;\, x \in [l_x..h_x'],\ y \in [l_y'..h_y] \rangle$

where $h_x' = \min(h_x, h_y - 1)$, $l_y' = \max(l_y, l_x + 1)$.

Exercise

Constraint propagation: Reduce a Domain

More generally:

$\langle x < y \ ; \ x \in [l_x..h_x], \ y \in [l_y..h_y] \rangle$

$\langle x < y \ ; \ x \in [l_x..h_x'], \ y \in [l_y' \ ..h_y] \rangle$

where $h_x' = \min(h_x, h_y - 1)$, $l_y' = \max(l_y, l_x+1)$.

Example: $\langle$x < y,y < z ; x ∈ [50..200],y ∈ [0..100],z ∈ [0..100]$\rangle$.

Apply above rule to x < y: ....

Apply it now to y < z: ....

Apply it again to x < y: ....

Exercise

Constraint propagation: Reduce a Domain

More generally:

$\langle x < y \; ; \; x \in [l_x..h_x], \; y \in [l_y..h_y] \rangle$

$\langle x < y \; ; \; x \in [l_x..h_x'], \; y \in [l_y' \, ..h_y] \rangle$

where $h_x' = \min(h_x, h_y - 1)$, $l_y' = \max(l_y, l_x + 1)$.

Example: $\langle x < y, y < z \; ; \; x \in [50..200], y \in [0..100], z \in [0..100] \rangle$.

Apply above rule to $x < y$:

$\langle x < y, y < z \; ; \; x \in [50..99], y \in [51..100], z \in [0..100] \rangle$.

Apply it now to $y < z$:

$\langle x < y, y < z \; ; \; x \in [50..99], y \in [51..99], z \in [52..100] \rangle$.

Apply it again to $x < y$:

$\langle x < y, y < z \; ; \; x \in [50..98], y \in [51..99], z \in [52..100] \rangle$.

Constraint propagation: Reduce Constraints

- Usually by introducing new constraints.
- Transitivity of <

$$\frac{\langle x<y,\ \ y<z;\ DE\rangle}{\langle x<y,\ \ y<z,\ \ x<z;\ DE\rangle}$$

This rule introduces new constraint, $x < z$.

- Resolution rule.
  L is a literal $\bar{L}$ is literal opposite to L.
  Thus: $\overline{\neg x} := x$, $\bar{x} := \neg x$.
  C_1 and C_2 clauses (=disjunctions of literals)

$$\frac{\langle C_1 \vee L, C_2 \vee \bar{L};\ \ DE\rangle}{\langle C_1 \vee L, C_2 \vee \bar{L}, C_1 \vee C_2\ ;\ DE\rangle}$$

This rule introduces new constraint, clause $C_1 \vee C_2$.

Exercise
apply resolution rule

- CSP: $\langle x \vee y,\ \neg x \vee y \vee z,\ \neg x \vee \neg z;\ DE \rangle$

Exercise
apply resolution rule

- $<x \vee y,\ \neg x \vee y \vee z,\ \neg x \vee \neg z;\ DE>$
- *first L=x*
  $<x \vee y,\ \neg x \vee y \vee z,\ \neg x \vee \neg z,\ y \vee z;\ DE>$
- *then L=z*
  $<x \vee y,\ \neg x \vee y \vee z,\ \neg x \vee \neg z,\ y \vee z,\ \neg x \vee y;\ DE>$
- *again L = x*
  $<x \vee y,\ \neg x \vee y \vee z,\ \neg x \vee \neg z,\ y \vee z,\ \neg x \vee y,\ y\ ;\ DE>$
  This means we can derive y = true!!

Constraint Propagation Algorithms

- Deal with scheduling of atomic reduction steps.
- Try to avoid useless applications of atomic reduction steps
- Stopping criterion for general CSP's: a local consistency notion.
  (Projection rule corresponds to local consistency notion.)

Hyper-arc consistency:
For every constraint C and every variable x with domain D, each value for x from D participates in a solution to C.
In case of binary constraints: hyper-arc consistency = arc consistency

Arc-consistency
CSP consistent

- $\langle x<y,\ y<z,\ x<z;\ x \in [0..5],\ y \in [1..7],\ z \in [3..8]\rangle$
- $\langle x<y,\ y<z,\ x<z;\ x \in [0..5],\ y \in [0..7],\ z \in [3..8]\ \rangle$
- $\langle x <>y,\ y<>z,\ z<>x;\ x \in [1..2],\ y \in [1..2],\ z \in [1..2]\rangle$
- Arc-consistent $\Rightarrow$ consistent CSP ??
- consistent CSP $\Rightarrow$ Arc-consistent ??

Example: Boolean Constraints

Happy: found all solutions.

Desired syntactic form (for preprocessing):

CNF

Constraint propagation:

$\langle x \wedge y = z;\ x \in D_x, y \in D_y, z \in \{1\} \rangle$

$\langle ;\ x \in D_x \cap \{1\}, y \in D_y \cap \{1\}, z \in \{1\} \rangle$ Write as

$x \wedge y = z,\ z = 1 \rightarrow x = 1,\ y = 1.$

Boolean Constraints

Atomic: if no domain of it contains more than one element

Split:

- Choose the most constrained variable.
- Apply the labelling rule:

$$\frac{x \in \{0,1\}}{x \in \{0\} \mid x \in \{1\}}$$

Proceed by cases: backtrack.

Boolean Constraints

- choice of specific domain reduction steps
- here: if the values of some variables are determined, then values of some other variables can be determined.

Example:

- x & y = z, we know z is true -->
  x is true and y is true