Consistency Enforcing and Constraint Propagation: Node and Arc Consistency

Solution Techniques

Consistency Enforcing and Constraint Propagation

Arc-Consistency

Consistency Enforcing and Constraint Propagation: Node and Arc Consistency

Constraint Processing, R. Dechter

Sections 3.1, 3.2, 3.4 (briefly)

Summary

Consistency Enforcing and Constraint Propagation: Node and Arc Consistency

Solution Techniques

Consistency Enforcing and Constraint Propagation

Arc-Consistency

- Node consistency and Arc Consistency

Solution Techniques for Constraint Network

Solving Constraint Networks

- Inference:
  - Infer new constraints based on existing ones
  - Eliminate values from variables that do not meet constraints
- Search:
  - Look for a solution trying different values of variables
  - backtracking and similar approaches
  - local search

Backtracking

general ideas

- Choose a variable x
- list its domain values
- for each value add a constraint $x = v$ and recursively evaluate the rest of the problem

Local Consistency

general ideas

- Partial assignments can lead to constraint violations
  - We can evaluate a constraint as soon as all variables in its scope are assigned
- We can backtrack as soon as a constraint is not locally consistent

Inference and constraint propagation

Example (inference)

- Variables: $\{A, B, C\}$
- Domain: $\{0, 1\}$ or true,false
- Constraint: $\{A \implies B, C \implies A, C\}$
- Propagating the constraints we can infer $\{A, B\}$
- Similar reasoning if we know $\{\neg B\}$ holds

Consistency

Consistency Methods

- Approximation of inference
  - arc, path and i-consistency
- tighter networks $\Rightarrow$ more efficient search
- Partial assignments can be discarded earlier

Consistency approaches

consistency enforcing

- Given a partial solution of length $i-1$ we extend the solution to one more variable
- i consistency:
  - for any legal value for $i-1$ variables
  - we can find a legal value for any other connected variables.
- Arc-Consistency: from 1 variable to 2
- Path-Consistency: from 2 variables to 3
- A network that is i-consistent for $i = 1, \cdots, n$ is globally consistent

Consistency and computational issues

consistency and computation

- The higher is i the better a search algorithm will behave
- time and space cost to ensure i-consistency is exponential in i
- Trade-off addressed with experimental evaluation

Example

Example (Constraint Propagation)

- Variables: $\{X, Y, T, Z\}$, $D_i = 1, 2, 3$
- Constraints: $X < Y, Y = Z, T < Z, X < T$

Node consistency

Node consistency

- Variable x_i, Domain D_i
- x_i is node consistent if every value of its domain satisfy every unary constraint
- $\forall v \in D_i \; \forall C = \{< x >, R_{x_i}\} \; a \in R_{x_i}$

Constraint propagation

Constraint Propagation

- We modify the constraint network so that:
  - local consistency is satisfied (enforcing consistency)
  - solutions do not change (maintaning equivalence)

Constraint propagation for node consistency

CP for node consistency

- If a variable x_i is not node consistent:
  - remove all values from D_i that do not satisfy all unary constraints
  - $D_i' = D_i \setminus \{v | \exists C = \{< x_i >, R_{x_i}\} \wedge v \notin R_{x_i}\}$
- D_i' contains only values that satisfy all unary constraints (enforcing consistency)
- all removed values could not be part of any solution (maintaning equivalence)

Arc Consistency

Example (Arc consistency)

- Variables x,y with domains $D_x = D_y = \{1, 2, 3\}$.
- $C = \{< x, y >, R_{x,y} = x < y\}$
- D_x and D_y are not arc consistent with $R_{x,y}$
- $D'_x = \{1, 2\}$ $D'_y = \{2, 3\}$ are arc consistent
- $D''_x = \{1\}$ $D''_y = \{2\}$ are arc consistent but...

Constraint propagation for arc consistency

CP for arc consistency

- If a variable x_i is not arc consistent w.r.t. x_j:
  - remove all values from D_i that do not have a matching value in x_j
- D'_i contains only values that satisfy binary constraints (enforcing consistency)
- all removed values could not be part of any solution (maintaning equivalence)

Arc Consistency

Arc Consistency

- Network $\mathcal{R} =< X, D, C >$
- $x_i, x_j \in X$
- x_i arc consitent w.r.t. x_j iff
  - $\forall a_i \in D_i \; \exists a_j \in D_j | (a_i, a_j) \in R_{x_i,x_j}$
- R_{x_i,x_j} is arc consistent iff x_i arc consistent w.r.t. x_j and x_j arc consistent w.r.t. x_i
- $\mathcal{R}$ is arc consitent iff all its constraints are arc consitent

Revise Procedure

Revise proc.

Algorithm 1 Revise((x_i),x_j)

Require: R_{x_i,x_j}, D_i, D_j

Ensure: D_i such that x_i is arc consistent w.r.t. x_j

```
for all a_i ∈ D_i do
    if ¬∃ a_j ∈ D_j | (a_i, a_j) ∈ R_{x_i,x_j} then
        delete a_i from D_i
    end if
end for
```

Equivalent to $D_i \leftarrow D_i \cap \pi_i(R_{ij} \bowtie D_j)$

Revise Procedure for Networks

Revise for Network

for all Pairs x_i, x_j that participate in a constraint **do**
    Revise((x_i),x_j);
    Revise((x_j),x_i);
end for

- This algorithm does not work!
- Revising arc consistency on a variable might make another variable not-arc consistent

Revising Networks

Example (Revise for Network)

- Variables x,y,z with domains $D_x\{0,1,2,3\}$, $D_y = \{1,2\}$, $D_z = \{0,1,2\}$.
- $C_{x,y} = \{< x, y >, R_{x,y} = x < y\}$, $C_{z,x} = \{< z, x >, R_{z,x} = z < x\}$

Revising Networks

An algorithm that does work!

AC-1

Require: $\mathcal{R} =< X, D, C >$
Ensure: $\mathcal{R}'$ the loosest arc consistent network for $\mathcal{R}$

```
repeat
    for all Pairs x_i, x_j that participate in a constraint do
        Revise((x_i),x_j);
        Revise((x_j),x_i);
    end for
until no domain is changed
```

- This algorithm does work!

Inconsistent Networks

AC-1 always terminate

- If we do not change any domain then we stop and $\mathcal{R}$ is AC
- If we remove a value we make at least one domain smaller
- If a domain is empty the network is inconsistent: we can not find any solution

Inconsistent Networks: Example

Example

Example

- Variables: $\{x, y, z\}$, domains $D_x = D_y = D_z = \{1, 2, 3\}$
- Constraints $\{x < y, y < z, z < x\}$
- apply AC-1

Computational complexity of AC-1

Comp. complexity

AC-1 is $O(nek^3)$

- n: nodes, e: edges, k: max number of values of a domain
- each cycle: $2ek^2$ operations
- worst case we delete 1 element from one domain at each cycle
- we can have at most nk cycles

Improving AC-1: AC-3

AC-3

Require: $\mathcal{R} =< X, D, C >$
Ensure: $\mathcal{R}'$ the loosest arc consistent network for $\mathcal{R}$

```
for all pairs (x_i, x_j) that participate in a constraint R_{x_i,x_j} ∈ R do
    Q ← Q ∪ {(x_i, x_j), (x_j, x_i)}
end for
while Q ≠ { } do
    pop (x_i, x_j) from Q
    REVISE((x_i),x_j)
    if D_i changed then
        Q ← Q ∪ {(x_k, x_i), k ≠ i, k ≠ j}
    end if
end while
```

AC-3 Example

Example

AC-3

- Variables x, y, z, domains $D_x = D_z = \{2, 5\}$, $D_y = \{2, 4\}$
- Constraints: $R_{x,z} = \{a_x, a_z, |(a_x \ mod \ a_z = 0)\}$
  $R_{y,z} = \{a_y, a_z, |(a_y \ mod \ a_z = 0)\}$
- Run AC-3

AC-3 Computational Complexity

Comp. Complexity

- $O(ek^3)$
- Revise for each couple is $O(k^2)$
- worst case we evaluate $2ek$
- because we can put back each couple at most k times

Consistency and Arc Consistency

Consistency Enforcing and Constraint Propagation: Node and Arc Consistency

Solution Techniques

Consistency Enforcing and Constraint Propagation

Arc-Consistency

Empty Domain and Arc Consistency

- Arc consistency + empty domain $\rightarrow$ inconsistent problem
- Arc consistent + all domains are not empty $\not\rightarrow$ consistent problem
- Arc consistency is not complete
  - It checks only single (binary) constraints and single domain constraint

Example: incompleteness of AC for consistency

Consistency Enforcing and Constraint Propagation: Node and Arc Consistency

Solution Techniques

Consistency Enforcing and Constraint Propagation

Arc-Consistency

Example

Binary Graph Colouring

- Variables: x, y, z Domain: $D_i = \{R, Y\}$
- Constraints: $x! = y, y! = z, z! = x$

Inconsistency

Inconsistencies when forcing consistency

- When forcing local consistency we can find out that the problem is inconsistent (e.g., arc consitency and empty domain).
- The opposite is not always true... but it is true in some cases
- For this class of problems local consistency ensures consistency of the problem: tractable cases
- Tractable because they are polynomial

Not tractable problem: example

Local consistent problem that is inconsistent

- Variables: x_1, x_2, x_3, x_4 Domain: $D = \{0, 1, 2\}$
- Constraints
  $x_1 \neq x_2, x_1 \neq x_3, x_1 \neq x_4, x_2 \neq x_3, x_2 \neq x_4, x_3 \neq x_4$
- For every value of every variable (e.g., 0) there is always a different value for another variable (e.g., 2) (arc consistent)
- For every couple of values of two variables (e.g., 0,1) there is always another value of another variable (e.g., 2)
- But we can not find 4 values that are all different in the domain $\{0, 1, 2\}$

Arc Consistency and Consistency

Why we have local consistency but global inconsistency

- Consider a tree.
- If each node is arc consistent with its children then the problem is arc consistent
- The problem is also globally consistent
- This is because siblings will never introduce inconsistency
- Cycles are the problem

Complete case for Arc Consistency

completeness for arc consistency

An arc (and node) consistent problem is globally consistent iff

- no empty domain
- only binary constraints
- primal graph contains no cycle

Solution algorithm for this type of problems

- Enforce arc consistency
- Note: no constraint addition $\rightarrow$ still acyclic
- If no domain is empty
  - Choose a node
  - Choose a value for the node and extend it to all its children
  - Propagate the choise value propagation
- Otherwise the problem is inconsistent

i-consistency for network

i-consistent $\mathcal{R}$

- $\mathcal{R}$ is i-consistent iff:
- for any consistent instantiation of $i - 1$ distinct variables
- there is a value of the ith values
- such that the i values satisfy all constraints among them

i-consistency example

Example

4-Queens problem

- The 4-Queen problem is 2-consistent
- The 4-Queen problem is not 3-consistent
- The 4-Queen problem is not 4-consistent

strong i-consistency for network

strong i-consistent $\mathcal{R}$

- $\mathcal{R}$ is **strong** i-consistent iff: $\mathcal{R}$ is j-consistent for any $j \leq i$
- If $\mathcal{R}$ is **strong** n-consistent then it is **globally** consistent
- For a globally consistent network we can extend any consistent partial instantiation to a complete instantiation without dead end: **backtrack free**

Exercise

Exercise 1

Consider the following network:

- Variables: {X,Y,Z,W}, Domain $D_i = \{0, 1, 2\}$
- Constraints: $X < Y$, $Z = X$, $Z < W$, $W < Y$

describe an execution of AC-3. Is the resulting network arc consistent ? Is the resulting network consistent ? Motivate your answers.