Consistency Enforcing and Constraint Propagation: Path consistency and i-consistency

Summary

- Path Consistency
- i-consistency

Path consistency

Basic Concepts

- $x, y, z, R_{x,y}, R_{y,z}, R_{x,z}$
- Arc consistency: every consistent value of x can be extended to y
- Path consistency every consistent couple of values for x, y can be extended to z

Path consistency: Example

Example (Path Consistency)

- $x, y, z,\ D = \{1, 2\}$
- $R_{x,y} = \{(1,1)(2,2)\}, R_{x,z} = \{(1,1)(1,2)(2,1)\}, R_{y,z} = \{(1,1)(2,2)\}$

Path consistency: Definition

Definition (Path Consistency for x, y relative to z)

- Couple of variables x, y and a third variable z
- Constraints $R_{x,y}, R_{x,z}, R_{y,z}$ (if a constraint does not exists all values are possible)
- x, y is path consistent w.r.t. z iff:
  - $\forall < a, b >\in R_{x,y} \wedge a \in D_x \wedge y \in D_y$
  - $\exists c \in D_z| < a, c >\in R_{x,z} \wedge < b, c >\in R_{y,z}$
- Graphically: a triangle in the matching diagram

Why Path consistency is important

importance of path consistency

If path consistency does not hold for $x = a, y = b$ relative to z then $< a, b >$ can not be part of any solution

- If we fix a, b we can not find any value for z
- But we need to assign z and the solution must satisfy all constraints

Path consistency for problems

Path Consistency for $\mathcal{R}$

$\mathcal{R}$ path consistent iff

- for every couple of variables x, y and every other variable z
- x, y path consistent relative to z

Path Consistency for $R_{x,y}$ relative to z

Constraint $R_{x,y}$ is path consistent relative to z

- every couple in $R_{x,y}$ is path consistent relative to z

Enforcing path consistency

Enforcing Path Consistency for $\mathcal{R}$

If $\mathcal{R}$ is not path consistent

- exists a couple of variables $x = a, y = b$ that can not be extended to z
- $x = a, y = b$ can not be part of any solution
- but we can not remove a, b from their respective domains!

Enforcing path consistency: Example

Example (Enforcing Path Consistency)

- Variables: x, y, z Domains: $D = \{1, 2\}$
- Constraints $R_{x,y} = \{(1,1)(1,2)(2,1)\}, R_{x,z} = \{(1,1)(2,1)(2,2)\}, R_{y,z} = \{(1,2)(2,1)(2,2)\}$
- $x = 1, y = 1$ can not be extended to any value of z but we have solutions with $x = 1$ and $y = 1$
- Removing $x = 1$ (or $y = 1$) could make other solutions disappear

Enforcing path consistency

Enforcing Path Consistency for $\mathcal{R}$

Given $x = a, y = b$ not path consistent relative to z

- we eliminate $< a, b >$ from $R_{x,y}$
- $R_{x,y}$ now is path consistent with repsect to z
- we did not remove solutions:
  - a, b could not be in any solution
  - $a \in D_x$ and $b \in D_y$ can still be used for other solutions
- Simpler problem: do not need to check z to realise a, b not a solution

Enforcing path consistency: Revise-3

Revise-3 proc.

Algorithm 1 Revise-3((x, y),z)

Require: A three variable subnetwork over x, y, z, $R_{x,y}, R_{y,z}, R_{x,z}$

Ensure: Revised $R_{x,y}$ path consistent relative to z

```
for all < a, b >∈ R_{x,y} do
    if ¬∃ c ∈ D_z|(a, c) ∈ R_{x,z} ∧ (b, c) ∈ R_{y,z} then
        delete a, b from R_{x,y}
    end if
end for
```

Equivalent to $R_{xy} \leftarrow R_{xy} \cap \pi_{xy}(R_{xz} \bowtie D_z \bowtie R_{zy})$

Revise-3: Example

Example (Revise-3)

- Variables: x, y, z Domains: $D = \{1, 2\}$
- Constraints $x! = y, y! = z, z! = x$
- Run Revise-3$((x, y), z)$

Inconsistent problem

Inconsistent Problem

If a revise makes a relation ($R_{x,y}$) empty the problem is inconsistent

- We have to assign x and y
- Every possible assignment will not satisfy $R_{x,y}$

node/arc consistency	remove values from domains	empty domain $\rightarrow$ inconsistency
path consistency	remove values from relations	empty relation $\rightarrow$ inconsistency

Loosing Path Consistency

PC-1

- Removing values from $R_{x,y}$
- We can loose pathconsistency wherever it depends on $R_{x,y}$
  - between couple <x,z> and y
  - between couple <y,z> and x

Path Consistency Algorithm

PC-1

Require: $\mathcal{R} =< X, D, C >$

Ensure: A path consistent network equivalent to $\mathcal{R}$

repeat
  for all $k \leftarrow 1$ to n **do**
    for all $i, j \leftarrow 1$ to $n \mid j \neq k \wedge i \neq k \wedge i < j$ **do**
      Revise((x_i, x_j),x_k);
    end for
  end for
until no constraint is changed

PC-1: Example

Example (Enforcing Path Consistency)

- Variables: x_1, x_2, x_3 Domains: $D = \{1, 2\}$
- Constraints $x_1 != x_3, x_2 != x_3$

Creating new constraints

New constraints

- We can create a new constraint between x and y when they share a constraint with another variable z
- We can not have a constraint when two nodes are not connected
- We could have constraint creation even if variables are connected but not directly connected: multiple constraint creation.

Example

Example (New constraints)

- x, y, z; R_{xz}
  - Can we create a new constraint bewteen x, y ?
- x, y, z, w; R_{xz}, R_{zw}, R_{wy}
  - Can we create a new constraint bewteen x, y ?

PC-1 Computational Complexity

Comp. Complexity

- $O(n^5 k^5)$
- Revise for each triplets is $O(k^3)$
- Each cycle we revise at most $O(n^3)$ triplets
- Number of cycles is at most $O(n^2 k^2)$
  - Because at each cycle we remove at least one element from one constraint, number of elements in all constraints is $O(n^2 k^2)$

Improvements for PC-1

Improve PC-1: PC-2

- Can do better than PC-1 (similarly to AC-1)
- We can focus on triplets that might have changed (simlilarly to AC-3)
- If $R_{x,y}$ is changed we re-process all triplets involving x, y
- x, y, z with z any other variable

Path Consistency Algorithm

PC-2

Require: $\mathcal{R} =< X, D, C >$

Ensure: A path consistent network equivalent to $\mathcal{R}$

$Q \leftarrow \{< i, k, j > | 1 \leq i < j \leq n \ \wedge 1 \leq k \leq n \ \wedge k \neq i \wedge k \neq j\}$

while $Q \neq \{\,\}$ **do**

  pop $< i, k, j >$ from Q

  Revise$((x_i, x_j), x_k)$;

  if R_{x_i, x_j} changed **then**

    $Q \leftarrow Q \cup \{< l, i, j >, < l, j, i > | 1 \leq l \leq n \ \wedge l \neq i \wedge l \neq j\}$

  end if

end while

PC-2 Computational Complexity

Comp. Complexity

- $O(n^3 k^5)$
- Revise for each triplets is $O(k^3)$
- Number of times we process the queue is at most $O(n^3 k^2)$
- Because we can put back an element at most $O(k^2)$

PC-2: Example

Example (Enforcing Path Consistency)

- Variables: x_1, x_2, x_3, x_4 Domains: $D = \{1, 2\}$
- Constraints $x_1 \neq x_2, x_2 \neq x_3, x_3 \neq x_4, x_4 \neq x_1$

Path consistency: alternative definition

Path consistency

- R_{x_i,x_j} is path consistent relative to a path of length m if we can find a sequence of other $m-2$ values such that all constraints along the path $i, i_1, \cdots, i_{m-1}, j$ are satisied.
- if we consider complete graphs the two definitions are the same
- if we consider only real path definitions are different

Inconsistency

Inconsistencies when forcing consistency

- When forcing arc or path consistency we can make a domain or a constraint empty, then the problem is inconsistent.
- The opposite is not always true... but it is true in some cases
- For this class of problems arc/path consistency ensures consistency of the problem: tractable cases
- Tractable because they are polynomial

Not tractable problem: example

arc/path 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
- Recall: 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

Complete case for Path Consistency

completeness for path consistency

- A path (and arc and node) consistent problem is consistent iff
  - no empty relation or empty domain
  - only binary constraints
  - only two values in the domain
- Even if primal graph has cycles

i-consistency: general concept

Basic ideas

- arc consistency consider sub-network of size 2
- path consistency consider sub-network of size 3
- i-consistency generalisation of this concept for sub-network of size $i - 1$

i-consistency

i-consistency

- Given relation R_S and a variable y
- $S \subseteq X$, $|S| = i - 1$, $y \notin S$
- R_S is i-consistent relative to y iff
  - for every tuple $t \in R_S$ there is a value $a_y \in D_y$ such that
  - (t, a) is consistent

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 strongly 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

strong i-consistency example

Example

all 5 equals

- Variables X con $|X| >= 5$; Domain: $D = \{0, 1\}$
- Constraints: allEquals on all subset of exactly 5 variables
- This problem is 6-consistent but not 5-consistent

global consistency

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 strongly n-consistent then it is globally consistent

enforcing i-consistency

adding constraints to enforce i-consistency

- To enforce i-consistency we might need to add constraints of $i - 1$ variables
- These constraints record forbidden combinations or no-good
- Therefore binary network might become non-binary
- Example: 4-Queens

Enforcing i-consistency: Revise-i

Revise-i proc.

Algorithm 2 Revise-i($(x_1, x_2, \cdots, x_{i-1})$,$x_i$)

Require: A network $\mathcal{R}$, a constraint R_S, which might be universal

Ensure: A constraint $R_S(S = x_1, x_2, \cdots, x_{i-1})$ which is i-consistent relative to x_i

```
for all instantiations ā_{i-1} ∈ R_S do
    if ¬∃ a_i ∈ D_i | (ā_{i-1}, a_i) is consistent then
        delete ā_{i-1} from R_S
    end if
end for
```

Complexity of Revise-i

Revise-i complexity

- Complexity of Revise-i for only binary constraints is $O(k^i)$
- With general constraint we have $O(2k^i)$
- We might need to check 2^i constraints
- If e bounds the constraints then $O(ek^i)$

i-Consistency Algorithm

i-consistency for networks

Require: $\mathcal{R} =< X, D, C >$
Ensure: An i-consistent network equivalent to $\mathcal{R}$
  repeat
    for all $S \subseteq X$ of size $i - 1$ **do**
      for all x_i **do**
        Revise-i((S),x_i);
      end for
    end for
  until no constraint is changed

3-Consistency and Path-Consistency

3-Consistency vs. Path Consistency

- On a binary network 3-Consistency = Path-Consistency
- If we have ternary constraints then not the same
- Example:
  - x, y, z $R_{x,y,z} = (0,0,0)$
  - Path consistency will do nothing: no binary constraint
  - 3-Consistency: at least add the constraint $R_{x,y} = < x, 0 >< y, 0 >$