

Problem Reduction for Constraint Satisfaction

Motivations:

1. Reduce problem to easier problems
2. Detect unsatisfiability

Edward Tsang (Copyright)

35

Monday, 04 February 2013

Node-consistency (NC), Definition

- Notation: C_x – constraint on variable x
- A CSP is *node-consistent* iff for all variables all values in its domain satisfy the constraints on that variable
- Note:
 - The statement “ P implies Q ” is *true* if P is *false*
 - Hence if all domains are empty, the problem is *node-consistent* (though *unsatisfiable*)

Edward Tsang (Copyright)

38



Monday, 04 February 2013

Problem Reduction Overview

- [Node-consistency \(NC\)](#)
- [Arc-consistency \(AC\) \(example\)](#)
- [Path-consistency \(PC\) \(example\)](#)
- [\(strong\) \$k\$ -consistency](#)
- [Directional Arc-consistency \(DAC\)](#)
- Other Consistency Properties (not covered)
 - Directional PC, Adaptive consistency, ...

Edward Tsang (Copyright)

36



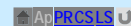
Monday, 04 February 2013

Arc-consistency (AC), Definition

- Notation $C_{x,y}$: constraint on variables x & y
- An *arc* (x, y) is *arc-consistent* iff
 - for every value a in the D_x which satisfies C_x ,
 - there exists at least one value b in D_y
 - such that $\langle y, b \rangle$ is compatible with $\langle x, a \rangle$
 - In this case, we say $\langle y, b \rangle$ *supports* $\langle x, a \rangle$
- A *problem* is *arc-consistent* iff every arc in its constraint graph is arc-consistent

Edward Tsang (Copyright)

39



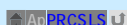
Monday, 04 February 2013

Node-consistency

- Principle of NC:
 - If unary constraint C_x exists for variable x
 - Remove any value v from D_x if v does not satisfy C_x
- E.g. D_x : days of the week;
 - C_x : this job must be done in weekdays
- Simple pre-processing strategy
- Cheap to compute: $O(n)$

Edward Tsang (Copyright)

37



Monday, 04 February 2013

Maintaining AC – Example

- Variables: x, y, z
- Domains: $\{1, 2, 3, 4\}$
- Constraints:
 - $x < y$; $y < z$

x	1	2	3	4
y	1	2	3	4
z	1	2	3	4

- $x < y$ means $\langle x, 4 \rangle$ not supported by y and $\langle y, 1 \rangle$ not supported by x
- $y < z$ means $\langle y, 4 \rangle$ not supported by z and $\langle z, 1 \rangle$ & $\langle z, 2 \rangle$ not supported by y
- Re-check $x < y$ would delete $\langle x, 3 \rangle$ as now (with $\langle y, 4 \rangle$ gone) it has no support from y

Edward Tsang (Copyright)

40



Monday, 04 February 2013

Maintaining Arc-consistency

- Principle of Arc-consistency:
 - If any variable y has no value to support $\langle x, v \rangle$
 - Then remove v from D_x
- Naïve algorithms:
 - Repeat constraint propagation
 - Until no more values can be removed
- Advanced algorithms: **record supports**
 - In order to focus on what to propagate
 - Complex data structure required

AC-4 Data Structure, Example

- Variables: x, y, z
- Domains: $\{1, 2, 3, 4\}$
- Constraints: $x < y; y < z$
- Data Structure:
 - $S_{\langle x,3 \rangle} = \{ \langle y,4 \rangle \}$
 - $S_{\langle x,4 \rangle} = \{ \}$
 - $S_{\langle y,2 \rangle} = \{ \langle x,1 \rangle, \langle z,3 \rangle, \langle z,4 \rangle \}$
 - $S_{\langle y,4 \rangle} = \{ \langle x,1 \rangle, \langle x,2 \rangle, \langle x,3 \rangle \}$
 - ...
- When $\langle y,4 \rangle$ is removed:
 - $S_{\langle y,4 \rangle}$ points to $\langle x,1 \rangle, \langle x,2 \rangle, \langle x,3 \rangle$
 - Counter[$\langle x,y \rangle, 3$] reduced to 0
 - $\langle x,3 \rangle$ is removed

x	1	2	3	4
y	1	2	3	4
z	1	2	3	4

Counter[$\langle x,y \rangle, 1$] = 3
 Counter[$\langle x,y \rangle, 2$] = 2
 Counter[$\langle x,y \rangle, 3$] = 1
 Counter[$\langle x,y \rangle, 4$] = 0
 Counter[$\langle y,x \rangle, 4$] = ? exercise

Arc-consistency Algorithms

- AC maintenance, lots of research
 - AC-1: Naïve but good for parallel processing
 - AC-4: Complex, using the concept of support
- Complexity manageable: $O(a^3ne)$ to $O(a^2e)$
- MAC: seen to be generally practical
 - Keep data structure to record supports permanently
 - No need to maintain AC from scratch

Remarks on AC

- AC has been maintained in this problem
- But it doesn't mean that all combinations of the remaining values are compatible with each other.
- All it means is every remaining value is supported by at least one value in every other variable

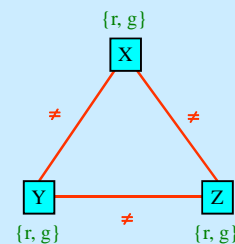
x	1	2	3	4
y	1	2	3	4
z	1	2	3	4

Algorithm AC-4

- Efficient algorithm for maintaining AC
 - Use data structure to reduce number of checks
- Support $S_{\langle x,a \rangle}$ records the set of labels that $\langle x,a \rangle$ supports
 - When $\langle x,a \rangle$ is removed, all labels in $S_{\langle x,a \rangle}$ lose support from $\langle x,a \rangle$; they need re-examination
- Counter[$\langle x, y \rangle, a$] records the number of supports that y provides to $\langle x,a \rangle$
 - When counter reduced to 0, remove $\langle x,a \rangle$
- $M[x,a] = 1$ if $\langle x,a \rangle$ has been rejected; 0 otherwise

Example: AC but Unsatisfiable

- This problem is AC
- But unsatisfiable
- Can we detect unsatisfiability?



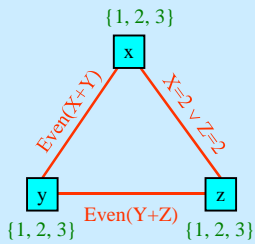
Path-consistency

- Principle:
 - Tightening constraints
 - By **constraint composition**
- Algorithms: PC-1, ..., PC-4
- Complexity manageable: $O(a^3n^3)$
- Is effort justifiable?

Constraint Composition

- Constraint C_{xy} & C_{yz} could tighten C_{xz}
- $C_{xz} = C_{xz} \wedge C_{xy} * C_{yz}$

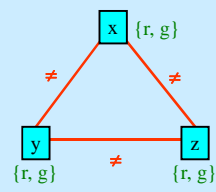
$$= \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix} \wedge \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix} \wedge \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$


Path-consistency (PC) Definition

- Notation C_S – the set of all relevant constraints on the set of variables S
- An **path** (x, y, z) is **path-consistent** iff
 - for every 2-compound label $\langle x, a \rangle \langle z, c \rangle$ that satisfies $C_{\{x,z\}}$,
 - there exists a value b in D_y
 - such that $\langle x, a \rangle \langle y, b \rangle \langle z, c \rangle$ satisfies $C_{\{x,y,z\}}$
- A **problem** is **path-consistent** iff every path in its constraint graph is path-consistent

PC Detecting Unsatisfiability



- Given $\langle x, r \rangle \langle y, g \rangle$, which satisfies $C_{x,y}$
- It has no compatible value in z
- Hence not PC

$$C_{x,y} = C_{y,z} = C_{x,z} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

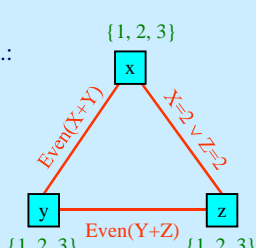
$$C_{x,y} = C_{x,y} \wedge C_{x,z} * C_{z,y}$$

$$= \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \wedge \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} * \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \wedge \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

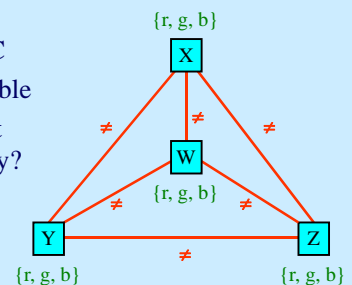
Binary Constraint Representation

- A Binary constraint may be represented by a matrix, e.g.:
- $D_x = D_y = D_z = \{1, 2, 3\}$
- C_{xy} : $X + Y$ must be even
- C_{xz} : at least one of X and Z must be equal to 2

$$C_{x,y} = C_{y,x} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \quad C_{x,z} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$


Example: PC but Unsatisfiable

- Problem is PC
- But unsatisfiable
- Can we detect unsatisfiability?



k-consistency, Definition

- A problem is **k-consistent** iff:
 - For all (k-1)-compound labels $(\langle x_1, v_1 \rangle \langle x_2, v_2 \rangle \dots \langle x_{k-1}, v_{k-1} \rangle)$
 - that satisfies all constraints on x_1, x_2, \dots, x_{k-1}
 - For every k^{th} variable x_k
 - There exists a value v_k such that all constraints on $x_1, x_2, \dots, x_{k-1}, x_k$ are satisfied
- Note: this is a definition, not every problem is k-C*

Directional-arc-consistency

- Observations:
 - Variables ordering is often static in a search
 - AC is bi-directional
- Principle:
 - Given an ordering of the variables
 - only remove v_x from D_x when
 - support does not exist for any **future** variable y
- Practical, may sit in other algorithms

Special Cases of k-Consistency

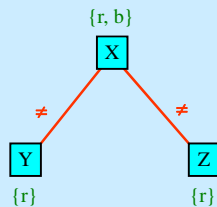
- AC \equiv 2-Consistency
- For binary problems, PC \equiv 3-Consistency

DAC Algorithm

- Given ordering of n variables, x_1, x_2, \dots, x_n
- For $k = n$ to 1 by -1 DO
 - For each variable x_j where $j < k$ & $C_{k,j}$ DO
 - Remove from domain of x_j any value not supported by x_k
- Each constraint is checked **once** only
 - Normally cheaper than maintaining AC

Strong k-consistency

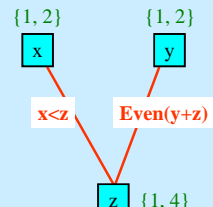
- A problem is **strong k-consistent** iff
 - it is 1-C, 2-C, ..., k-C
- Complexity grows with k



This problem is 3-C but not 2-C!

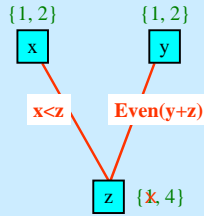
Maintaining DAC, Example 1

- Under order $(x \rightarrow y \rightarrow z)$
- Examine $y \rightarrow z$ (even sum)
 - $\langle y, 1 \rangle$ supported by $\langle z, 1 \rangle$
 - $\langle y, 2 \rangle$ supported by $\langle z, 4 \rangle$
- Examine $x \rightarrow z$ ($x < z$)
 - $\langle x, 1 \rangle$ supported by $\langle z, 4 \rangle$
 - $\langle x, 2 \rangle$ supported by $\langle z, 4 \rangle$
- Examine $x \rightarrow y$
 - No constraint



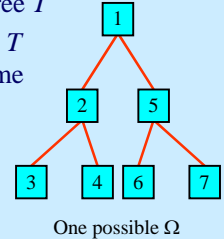
Maintaining DAC, Example 2

- Under order ($z \rightarrow y \rightarrow x$)
- Examine $y \rightarrow x$
No constraint
- Examine $z \rightarrow x$ ($x < z$)
 $\langle z, 1 \rangle$ not supported by x
 $\langle z, 4 \rangle$ supported by $\langle x, 1 \rangle$ and $\langle x, 2 \rangle$
- Examine $z \rightarrow y$ (even sum)
 $\langle z, 1 \rangle$ already deleted
 $\langle z, 4 \rangle$ supported by $\langle y, 2 \rangle$



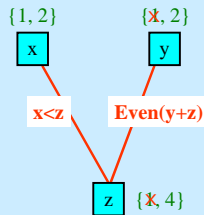
Tree-search Algorithm

- If the constraint graph is a tree T
- Ω : Ordering the variables in T such that parents always come before children
- Maintain DAC under Ω
- Then search under Ω
– Such search is guaranteed *backtrack-free!*



Maintaining DAC, Example 3

- Under order ($y \rightarrow z \rightarrow x$)
- Examine $z \rightarrow x$ ($x < z$)
 $\langle z, 1 \rangle$ not supported by x
 $\langle z, 4 \rangle$ supported by $\langle x, 1 \rangle, \langle x, 2 \rangle$
- Examine $y \rightarrow x$ (no constraint)
- Examine $y \rightarrow z$ (even sum)
 $\langle y, 1 \rangle$ not supported by z
– as $\langle z, 1 \rangle$ has been removed
 $\langle y, 2 \rangle$ supported by $\langle z, 4 \rangle$



Problem Reduction, Summary

- Aim: reduce problem to one that is easier to solve, or detect dead-ends
- Concept is simple, procedures could be complex
- General Concept: k-consistency
– NC=1-C, AC=2-C, PC=3-C for binary CSPs
- NC & AC potentially reduce domains
- PC potentially tightens binary constraints
- k-C for $k > 2$ potentially tightens k-1 constraints

Maintaining DAC, Remarks

- To maintain AC, one must
– Remove $\langle z, 1 \rangle$ (no support from x)
– Remove $\langle y, 1 \rangle$ (no support from z after $\langle z, 1 \rangle$ is removed)
- Maintaining DAC under both ($x \rightarrow y \rightarrow z$) and ($z \rightarrow y \rightarrow x$) does not obtain AC
– Only $\langle z, 1 \rangle$ is removed
- Maintaining DAC under the right order ($y \rightarrow z \rightarrow x$) matters

