Constraint Propagation

Juhani Peltonen

February 11, 2009

Outline

Outline

Introduction to Constraint Propagation

- Constraint satisfaction problem (CSP) includes
  - a set of variables,
  - a domain for each variable, and
  - a set of constraints restricting combinations of values the variables can take.
- To solve a CSP: find a value for each variable such that all the constraints are satisfied.
- Constraint satisfaction problems have high computational complexity.
- Exploring the entire space is too expensive.

Introduction to Constraint Propagation

- The searched space can be reduced by ruling out values that cannot take part in any solution.
- This is constraint propagation.

Example (Constraint propagation)

- If $x_1, x_2 \in [1 \dots 10]$ and $|x_1 - x_2| > 5$,
- then $x_1, x_2 \notin \{5, 6\}$.

Outline

Basic Definitions

Definition (Constraint)

A constraint c is a relation defined on a sequence of variables $X(c) = (x_{i_1}, \ldots, x_{i_{|X(c)|}})$, called the scheme of c. c is the subset of $\mathbb{Z}^{|X(c)|}$ that contains the combinations of values (or tuples) $\tau \in \mathbb{Z}^{|X(c)|}$. $|X(c)|$ is the arity of c.

Example (Constraint)

- `alldifferent`$(x_1, x_2, x_3) \equiv (v_i \neq v_j \wedge v_i \neq v_k \wedge v_j \neq v_k)$: an infinite set of 3-tuples in $\mathbb{Z}^3$ such that all values are different
- $c(x_1, x_2, x_3) = \{(2,2,3),(2,3,2),(2,3,3),(3,2,2),(3,2,3),(3,3,2)\}$: a finite set of 3-tuples containing both values 2 and 3 and only them

Basic Definitions

Definition (Constraint network)

A constraint satisfaction problem is defined by a constraint network (or network). Network N is a tuple (X, D, C) and is composed of

- a finite sequence of integer variables $X = (x_1, \ldots, x_n)$,
- a domain for X, i.e., a set $D = D(x_1) \times \cdots \times D(x_n)$, where $D(x_i) \subset \mathbb{Z}$ is the finite set of values that variable x_i can take, and
- a set of constraints $C = \{c_1, \ldots, c_e\}$, where variables in $X(c_j)$ are in X.

In addition, a network N is

- normalized iff two different constraints in C_N do not involve exactly the same variables and
- binary iff for all $c_i \in C_N$, $|X(c_i)| = 2$.

Basic Definitions

Definition (Tightenings of a network)

The space $\mathcal{P}_N$ of all possible tightenings of a network $N = (X, D, C)$ is the set of networks $N' = (X, D', C')$ such that $D' \subseteq D$ and for all $c \in C$ there exists $c' \in C'$ with $X(c') = X(c)$ and $c' \subseteq c$.

Definition (Domain-based tightenings)

The space $\mathcal{P}_{ND}$ of domain-based tightenings of a network $N = (X, D, C)$ is the set of networks in $\mathcal{P}_N$ with the same constraints as N, i.e., $N' \in \mathcal{P}_{ND}$ iff $X_{N'} = X$, $D_{N'} \subseteq D$, and $C_{N'} = C$.

- The "interesting" tightenings are those which preserve the solutions.

Outline

Arc Consistency

Arc consistency

- is a technique to propagate constraints which
- guarantees every value in a domain to be consistent with every constraint.

Definition (Arc consistency)

Given a network $N = (X, D, C)$, a constraint $c \in C$, and a variable $x_i \in X(c)$,

- a value $v_i \in D(x_i)$ is consistent with c in D iff there exists a valid tuple τ satisfying c such that $v_i = \tau[x_i]$.
- The domain D is arc consistent on c for x_i iff all the values in $D(x_i)$ are consistent with c in D.
- The network N is arc consistent iff D is arc consistent for all variables in X on all constraint in C.

Arc Consistency

Example

Let N be a network with $X = (x_1, x_2, x_3)$,
$D(x_1) = D(x_2) = D(x_3) = \{1, 2, 3\}$, and
$c_{12} \equiv (x_1 = x_2)$ and $c_{23} \equiv (x_2 < x_3)$.

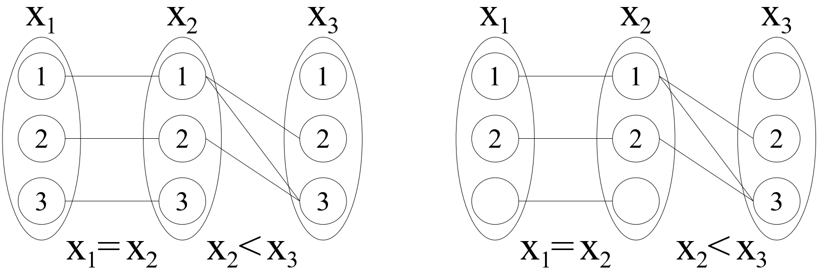


Figure: Network before arc consistency (left) and after (right)

Outline

Arc Consistency Algorithms: AC3

function *ac3*()

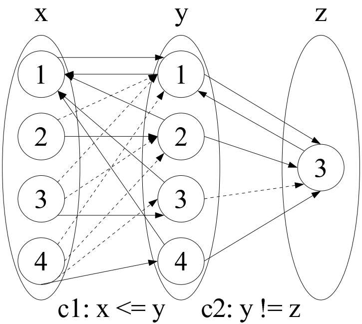
$Q = \{(x_i, c) \mid c \in C, x_i \in X(c)\}$
while $Q \neq \emptyset$ **do**
  select and remove (x_i, c) from Q
  if *revise*(x_i, c) **then**
    if $D(x_i) = \emptyset$ **then**
      return false
    else
      $Q = Q \cup \{(x_j, c') \mid c' \in C \wedge c' \neq c \wedge x_i, x_j \in X(c') \wedge j \neq i\}$
    end if
  end if
end while
return true

function *revise*(x_i, c)

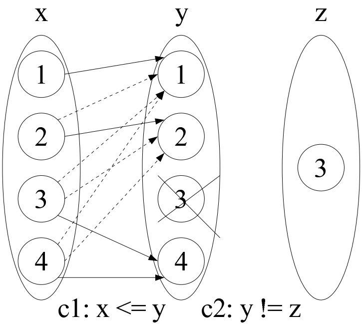
$CHANGE \leftarrow$ **false**
for all $v_i \in D(x_i)$ **do**
  if $\nexists \tau \in c \cap \pi_{X(c)}(D)$ with $\tau[x_i] = v_i$ **then**
    remove v_i from $D(x_i)$
    $CHANGE \leftarrow$ **true**
  end if
end for
return $CHANGE$

Arc Consistency Algorithms: AC3

Example



Q = {(x,c1), (y,c1), (y, c2), (z, c2)}



Q = {(x, c1)}

Outline

Summary

- A constraint satisfaction problem is defined by a constraint network.
- Constraint propagation is used to make a constraint satisfaction problem "easier" to solve by tightening the network.
- The idea of arc consistency is to enforce that all the values in the domains of variables are consistent with all the constraints.
- AC3 is one way to achieve arc consistency.

Thank you!
Questions?