DM826 – Spring 2011

Modeling and Solving Constrained Optimization Problems

Lecture 4

Constraint Propagation and Local Consistency

Marco Chiarandini

Department of Mathematics & Computer Science
University of Southern Denmark

CSP

Input:

- **Variables** $X = (x_1, \ldots, x_n)$
- **Domain Expression** $\mathcal{DE} = \{x_1 \in D(x_1), \ldots, x_n \in D(x_n)\}$

a constrained satisfaction problem (CSP) is

$$\mathcal{P} = \langle X, \mathcal{DE}, \mathcal{C} \rangle$$

$\mathcal{C}$ finite set of constraints each on a subsequence of X.
$C \in \mathcal{C}$ on $Y = (y_1, \ldots, y_k)$ is $C \subseteq D(y_1) \times \ldots \times D(y_k)$

$(v_1, \ldots, v_n) \in D(x_1) \times \ldots \times D(x_n)$ is a solution of $\mathcal{P}$
if for each constraint $C_i \in \mathcal{C}$ on $x_{i_1} \ldots, x_{i_m}$ it is

$$(v_{i_1}, \ldots, v_{i_m}) \in C_i$$

Notation and Terminology

Finite domains $\rightsquigarrow$ w.l.g. $D \subseteq \mathbf{Z}$

Constraint C: relation on a (ordered) *subsequence* of variables

- $X(C) = (x_{i_1}, \ldots, x_{i_{|X(C)|}})$ is the scheme or scope
- $|X(C)|$ is the arity of C (unary/binary/non-binary)
- $C \subseteq \mathbf{Z}^{|X(C)|}$ containing combinations of valid values (or tuples) $\tau \in \mathbf{Z}^{|X(C)|}$
- constraint check: testing whether a τ satisfies C
- $\mathcal{C}$: a t-tuple of constraints $\mathcal{C} = (C_1, \ldots, C_t)$
- expression
  - extensional: specifies satisying tuples (aka `table` in Comet)
  - intensional: specifies the characteristic function

CSP normalized: iff two different constraints do not involve exactly the same vars
CSP binary iff for all $C_i \in \mathcal{C}, |X(C)| = 2$

Notation and Terminology

Given a tuple τ on a sequence Y of variables and $W \subseteq Y$,

- $\tau[W]$ is the restriction of τ to variables in W (ordered accordingly)
- $\tau[x_i]$ is the value of x_i in τ
- $C \subseteq C'$ if $X(C) = X(C')$ and for all $\tau \in C$ the reordering of τ according to $X(C')$ satisfies C'.

Example

$C(x_1, x_2, x_3): \quad x_1 + x_2 = x_3$
$C'(x_1, x_2, x_3): \quad x_1 + x_2 \leq x_3$

$C \subseteq C'$

Given $Y \subseteq X(C)$, $\pi_Y(C)$ denotes the projection of C on Y. It contains tuples on Y that can be extended to a tuple on $X(C)$ satisfying C.

Notation and Terminology

Given $\mathcal{P} = \langle X, \mathcal{DE}, \mathcal{C} \rangle$ the instantiation I is a tuple on $Y = (x_1, \ldots, x_k) \subseteq X$: $((x_1, v_1), \ldots, (x_k, v_k))$

- I on Y is valid iff $\forall x_i \in Y$, $I[x_i] \in D(x_i)$
- I on Y is locally consistent on Y iff it is valid and for all $C \in \mathcal{C}$ with $X(C) \subseteq Y$, $I[X(C)]$ satisfies C
- a solution is an instantiation I on $X(C)$ which is locally consistent
- I on Y is globally consistent if it can be extended to a solution, i.e., there exists $s \in sol(|CSP)$ with $I = s[Y]$

Example

$$\mathcal{P} = \langle X = (x_1, x_2, x_3, x_4), \mathcal{DE} = \{D(x_i) = \{1..5\}, \forall i\},$$
$$\mathcal{C} = \{C_1 \equiv \texttt{alldiff}(x_1, x_2, x_3), C_2 \equiv x_1 \leq x_2 \leq x_3, C_3 \equiv x_4 \geq 2x_2\}\rangle$$

$\pi_{x_1, x_2}(C_1) \equiv (x_1 \neq x_2)$
$I_1 = ((x_1, 1), (x_2, 2), (x_4, 7))$ is not valid
$I_2 = ((x_1, 1), (x_2, 1), (x_4, 3))$ is local consistent since $X(C_3) \subseteq Y$ and $I_2[X(C_3)]$ satisfies C_3
I_2 is not global consistent: $sol(\mathcal{P}) = \{(1, 2, 3, 4), (1, 2, 3, 5)\}$

Notation and Terminology

CSP is NP-complete!
$\rightsquigarrow$ solved by extending partial instantiations to global consistent ones

$\mathcal{P}$' is a tightening of $\mathcal{P}$ if
$X = X_{\mathcal{P}'}, \mathcal{DE}_{\mathcal{P}'} \subseteq \mathcal{DE}, \forall C \in \mathcal{C}, \exists C' \in \mathcal{C}, X(C) = X(C')$ and $C' \subseteq C$.
Any instantiation I on $Y \subseteq X_{\mathcal{P}}$ locally inconsistent for $\mathcal{P}$ is locally inconsistent for $\mathcal{P}'$

$\mathcal{S}_{\mathcal{P}}$ is the space of all tightening for $\mathcal{P}$

We are interested in the tightenings that preserve the set of solutions ($\mathrm{sol}(\mathcal{P}) = \mathrm{sol}(\mathcal{P}')$) whose space is denoted $\mathcal{S}_{\mathcal{P}}^{\mathrm{sol}}$ and among them the smallest

$\mathcal{P}^* \in \mathcal{S}_{\mathcal{P}}^{\mathrm{sol}}$ is global consistent if any instantiation I on $Y \subseteq X$ which is locally consistent in $\mathcal{P}^*$ can be extended to a solution of $\mathcal{P}$.

Computing $\mathcal{P}^*$ is exponential in time and space $\rightsquigarrow$ search a close $\mathcal{P}$ in polynomial time and space

Define a property Φ that states necessary conditions on instantiations that enter in the definition of local consistency

Constraint Propagation

We reach a $\mathcal{P}$ that is Φ consistent by constraint propagation:

- tighten $\mathcal{DE}$
- tighten $\mathcal{C}$, ex: $x_1 + x_2 \leq x_3 \rightsquigarrow x_1 + x_2 = x_3$
- add C to $\mathcal{C}$

this is implemented by

- reduction rules: sufficient conditions to rule out values that have no chance to appear in a solution
- rules iterations: a set of reduction rules for each set of constraint that tighten

Focus on domanin-based tightenings

Domain-based tightenings

Task:
Finding a tightening $\mathcal{P}$ in $\mathcal{S}_{\mathcal{P}}$ such that:
forall $x_i \in X_{\mathcal{P}}$, $D_{\mathcal{P}'}(x_i)$ contains only values that belong to a solution itself,
i.e., $D_{\mathcal{P}'}(x_i) = \pi_{\{x_i\}}(\mathrm{sol}(\mathcal{P}))$

It is clearly NP-hard since it corresponds to solving $\mathcal{P}$ itself.

- Reduction rules:

  $D(x_i) \leftarrow D(x_i) \cap \{v_i | D(x_1) \times D(x_j - 1) \times \{v_i\} \times \ldots D(x_j + 1) \times \ldots D(x_k) \cap C \neq \emptyset\}$

- Rules iterations

Define Φ: e.g., unary, arc, path, k-consistency

Domain-based tightenings

Note: Not all Φ-consistent tightenings preserve the solutions
We search for the Φ-closure $\Phi(\mathcal{P})$ (the union of all $\mathcal{P}' \in \mathcal{S}_{\mathcal{P}}$ Φ-consistent) $\equiv$ enforcing Φ consistency

Example

$$\mathcal{P} = \langle X = (x_1, x_2, x_3, x_4), \mathcal{DE} = \{D(x_i) = \{1, 2\}, \forall i\},$$
$$\mathcal{C} = \{C_1 \equiv x_1 \leq x_2, C_2 \equiv x_2 \leq x_3, C_3 \equiv x_1 \neq x_3\}\rangle$$

Φ all values for all variables can be extended consistently to a second variable

$$\mathcal{P}' = \langle X = (x_1, x_2, x_3, x_4), \mathcal{DE} = \{D(x_1) = 1, D(x_2) = 1, D(x_3) = 2, \forall i\},$$
$$\mathcal{C} = \{C_1 \equiv x_1 \leq x_2, C_2 \equiv x_2 \leq x_3, C_3 \equiv x_1 \neq x_3\}\rangle$$

$\mathcal{P}$' is consitent but it does not contain $(1, 2, 2)$ which is in $\mathrm{sol}(\mathcal{P})$
$\Phi(\mathcal{P}) : \langle X, \mathcal{DE}_\Phi, \mathcal{C}\rangle$ $D_\Phi(x_1) = 1, D_\Phi(x_2) = \{1, 2\}, D_\Phi(x_3) = 2$

Domain-based tightenings

$\Phi(\mathcal{P})$ can be computed by a greedy algorithm:

Proposition (Fixed Point): If a domain based consistency property Φ is stable under union, then for any $\mathcal{P}$, the $\mathcal{P}$' with $\mathcal{DE}_{\mathcal{P}'}$ obtained by iteratively removing values that do not satisfy Φ until no such value exists is the Φ-closure of $\mathcal{P}$.

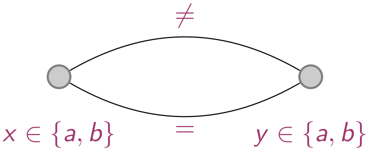
Generalized Arc Consistency (GAC)

Given $\mathcal{P}$, $C \in \mathcal{C}$, $x_i \in X(C)$

- $v \in D(x_i)$ is consistent with C in $\mathcal{DE}$ iff $\exists$ a valid tuple τ for C: $v_i = \tau[x_i]$. τ is called support for (x_i, v_i)
- $\mathcal{DE}$ is GAC on C for x_i iff all values in $D(x_i)$ are consistent with C in $\mathcal{DE}$ (i.e., $D(x_i) \subseteq \pi_{\{x_i\}}(C \cap \pi_{\{X(C)\}}(\mathcal{DE}))$)
- $\mathcal{P}$ is GAC iff $\mathcal{DE}$ is GAC for all v in X on all $C \in \mathcal{C}$
- $\mathcal{P}$ is arc inconsistent iff the only domain tighter than $\mathcal{DE}$ which is GAC for all variables on all constraints is the empty set.

In general arc consistency does not imply global consistency.

An arc consitent but inconsistent CSP:



A consistent but not arc consistent CSP:

AC3

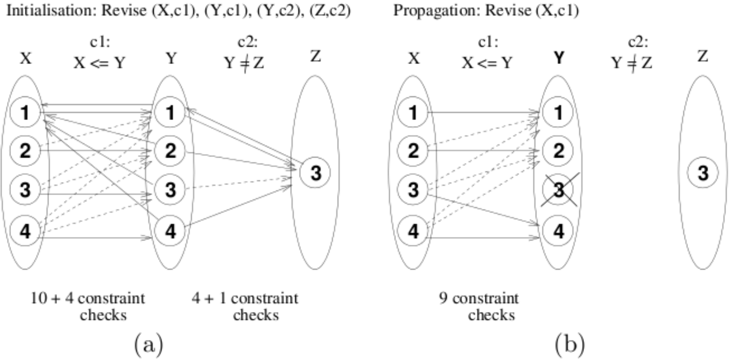
function Revise3(**in** x_i: variable; c: constraint)**: Boolean** ;
begin
CHANGE ← **false**;
foreach $v_i \in D(x_i)$ **do**
if $\nexists \tau \in c \cap \pi_{X(c)}(D)$ *with* $\tau[x_i] = v_i$ **then**
remove v_i from $D(x_i)$;
CHANGE ← **true**;
return CHANGE ;
end

function AC3/GAC3(**in** X: set)**: Boolean** ;
begin
/* initalisation */;
$Q \leftarrow \{(x_i, c) \mid c \in C, x_i \in X(c)\}$;
/* propagation */;
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 $Q \leftarrow Q \cup \{(x_j, c') \mid c' \in C \wedge c' \neq c \wedge x_i, x_j \in X(c') \wedge j \neq i\}$;
return **true** ;
end

$O(er^3d^r)$ time

AC3
Example

$$\mathcal{P} = \langle X = (x, y, z),\ \mathcal{DE} = \{D(x) = D(y) = \{1, 2, 3, 4\}, D(z) = \{3\}\},$$
$$\mathcal{C} = \{C_1 \equiv x \leq y, C_2 \equiv y \neq z\}\}\rangle$$



AC4

function `AC4`(**in** X: set): **Boolean** ;
begin
/* initialization */;
$Q \leftarrow \emptyset$; $S[x_j, v_j] = 0, \forall v_j \in D(x_j), \forall x_j \in X$;
foreach $x_i \in X, c_{ij} \in C, v_i \in D(x_i)$ **do**
initialize `counter`$[x_i, v_i, x_j]$ to $|\{v_j \in D(x_j) \mid (v_i, v_j) \in c_{ij}\}|$;
if *counter*$[x_i, v_i, x_j] = 0$ **then** remove v_i from $D(x_i)$ and add (x_i, v_i) to Q;
add (x_i, v_i) to each $S[x_j, v_j]$ s.t. $(v_i, v_j) \in c_{ij}$;
if $D(x_i) = \emptyset$ **then** return **false** ;
/* propagation */;
while $Q \neq \emptyset$ **do**
select and remove (x_j, v_j) from Q;
foreach $(x_i, v_i) \in S[x_j, v_j]$ **do**
if $v_i \in D(x_i)$ **then**
`counter`$[x_i, v_i, x_j]$ = `counter`$[x_i, v_i, x_j] - 1$;
if *counter*$[x_i, v_i, x_j] = 0$ **then**
remove v_i from $D(x_i)$; add (x_i, v_i) to Q;
if $D(x_i) = \emptyset$ **then** return **false** ;
return **true** ;
end

AC4

Example

$$\mathcal{P} = \langle X = (x, y, z),\ \mathcal{DE} = \{D(x) = D(y) = \{1,2,3,4\}, D(z) = \{3\}\},$$
$$\mathcal{C} = \{C_1 \equiv x \leq y, C_2 \equiv y \neq z\}\}\rangle$$

$$\begin{array}{lll}
\texttt{counter}[x,1,y] = 4 & \texttt{counter}[y,1,x] = 1 & \texttt{counter}[y,1,z] = 1 \\
\texttt{counter}[x,2,y] = 3 & \texttt{counter}[y,2,x] = 2 & \texttt{counter}[y,2,z] = 1 \\
\texttt{counter}[x,3,y] = 2 & \texttt{counter}[y,3,x] = 3 & \texttt{counter}[y,3,z] = 0 \\
\texttt{counter}[x,4,y] = 1 & \texttt{counter}[y,4,x] = 4 & \texttt{counter}[y,4,z] = 1 \\
 & & \texttt{counter}[z,3,y] = 3
\end{array}$$

$$\begin{array}{ll}
S[x,1] = \{(y,1),(y,2),(y,3),(y,4)\} & S[y,1] = \{(x,1),(z,3)\} \\
S[x,2] = \{(y,2),(y,3),(y,4)\} & S[y,2] = \{(x,1),(x,2),(z,3)\} \\
S[x,3] = \{(y,3),(y,4)\} & S[y,3] = \{(x,1),(x,2),(x,3)\} \\
S[x,4] = \{(y,4)\} & S[y,4] = \{(x,1),(x,2),(x,3),(x,4),(z,3)\} \\
 & S[z,3] = \{(y,1),(y,2),(y,4)\}
\end{array}$$

AC6

function `AC6`(**in** X: set): **Boolean** ;
  begin
    /* initialization */;
    $Q \leftarrow \emptyset$; $S[x_j, v_j] = 0, \forall v_j \in D(x_j), \forall x_j \in X$;
    foreach $x_i \in X, c_{ij} \in C, v_i \in D(x_i)$ **do**
      $v_j \leftarrow$ smallest value in $D(x_j)$ s.t. $(v_i, v_j) \in c_{ij}$;
      if v_j *exists* **then** add (x_i, v_i) to $S[x_j, v_j]$;
      else remove v_i from $D(x_i)$ and add (x_i, v_i) to Q;
      if $D(x_i) = \emptyset$ **then** return **false** ;
    /* propagation */;
    while $Q \neq \emptyset$ **do**
      select and remove (x_j, v_j) from Q;
      foreach $(x_i, v_i) \in S[x_j, v_j]$ **do**
        if $v_i \in D(x_i)$ **then**
          $v'_j \leftarrow$ smallest value in $D(x_j)$ greater than v_j s.t. $(v_i, v_j) \in c_{ij}$;
          if v'_j *exists* **then** add (x_i, v_i) to $S[x_j, v'_j]$;
          else
            remove v_i from $D(x_i)$; add (x_i, v_i) to Q;
            if $D(x_i) = \emptyset$ **then** return **false** ;
    return **true** ;
  end

AC6
Example

$$\mathcal{P} = \langle X = (x, y, z),\ \mathcal{DE} = \{D(x) = D(y) = \{1, 2, 3, 4\}, D(z) = \{3\}\},$$
$$\mathcal{C} = \{C_1 \equiv x \leq y, C_2 \equiv y \neq z\}\}\rangle$$

$S[x,1] = \{(y,1),(y,2),(y,3),(y,4)\}$
$S[x,2] = \{\}$
$S[x,3] = \{\}$
$S[x,4] = \{\}$

$S[y,1] = \{(x,1),(z,3)\}$
$S[y,2] = \{(x,2)\}$
$S[y,3] = \{(x,3)\}$
$S[y,4] = \{(x,4)\}$
$S[z,3] = \{(y,1),(y,2),(y,4)\}$

Reverse2001

function Revise2001(**in** x_i: variable; c_{ij}: constraint)**: Boolean** ;
  begin
    CHANGE $\leftarrow$ **false**;
    foreach $v_i \in D(x_i)$ *s.t.* *Last*$(x_i, v_i, x_j) \notin D(x_j)$ **do**
      $v_j \leftarrow$ smallest value in $D(x_j)$ greater than Last(x_i, v_i, x_j) s.t. $(v_i, v_j) \in c_{ij}$;
      if v_j *exists* **then** Last$(x_i, v_i, x_j) \leftarrow v_j$;
      else
        remove v_i from $D(x_i)$;
        CHANGE $\leftarrow$ **true**;
    return CHANGE ;
  end

function AC3/GAC3(**in** X: set)**: Boolean** ;
  begin
    /* initalisation */;
    $Q \leftarrow \{(x_i, c) \mid c \in C, x_i \in X(c)\}$;
    /* propagation */;
    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 $Q \leftarrow Q \cup \{(x_j, c') \mid c' \in C \wedge c' \neq c \wedge x_i, x_j \in X(c') \wedge j \neq i\}$;
    return **true** ;
  end

Reverse2001
Example

$$\mathcal{P} = \langle X = (x, y, z),\ \mathcal{DE} = \{D(x) = D(y) = \{1, 2, 3, 4\}, D(z) = \{3\}\},$$
$$\mathcal{C} = \{C_1 \equiv x \leq y, C_2 \equiv y \neq z\}\}\rangle$$

$\texttt{Last}[x, 1, y] = 1$	$\texttt{Last}[y, 1, x] = 1$	$\texttt{Last}[y, 1, z] = 3$
$\texttt{Last}[x, 2, y] = 2$	$\texttt{Last}[y, 2, x] = 1$	$\texttt{Last}[y, 2, z] = 3$
$\texttt{Last}[x, 3, y] = 3$	$\texttt{Last}[y, 3, x] = 1$	$\texttt{Last}[y, 3, z] = nil$
$\texttt{Last}[x, 4, y] = 4$	$\texttt{Last}[y, 4, x] = 1$	$\texttt{Last}[y, 4, z] = 3$
		$\texttt{Last}[z, 3, y] = 1$

References

Bessiere C. (2006). **Constraint propagation**. In *Handbook of Constraint Programming*, edited by F. Rossi, P. van Beek, and T. Walsh, chap. 3. Elsevier. Also as Technical Report LIRMM 06020, March 2006.