DM826 – Spring 2014

Modeling and Solving Constrained Optimization Problems

Lecture 5
Constraint Propagation and Local Consistency

Marco Chiarandini

Department of Mathematics & Computer Science
University of Southern Denmark

Outline

1. Definitions

2. Local Consistency

Reasoning with Constraints

Constraint Propagation, aka:

- constraint relaxation
- filtering algorithms
- narrowing algorithms
- constraint inference
- simplification algorithms
- label inference
- local consistency enforcing
- rules iteration
- proof rules

Local Consistency define properties that the constraint problem must satisfy *after* constraint propagation

Rules iteration defines properties on the process of propagation itself, that is is kind and order of operations of reduction applied to the problem

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 satisfying tuples (aka `table` or `extensional` via `DFA` or `TupleSet` in gecode).
    eg. $c(x_1, x_2) = \{(2,2), (2,3), (3,2), (3,3)\}$
  - intensional: specifies the characteristic function. eg. `alldiff`(x_1, x_2, x_3)

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

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

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
- if $X(C) = X(C')$ and $C \subseteq C'$ then for all $\tau \in C$ the reordering of τ according to $X(C')$ satisfies C'.

Example

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

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

$C \subseteq C'$

Notation and Terminology

- 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.
- given $X(C_1) = X(C_2)$, the intersection $C_1 \cap C_2$ contains the tuples τ that satisfy both C_1 and C_2
- join of $\{C_1 \dots C_k\}$ is the relation with scheme $\cup_{i=1}^{k} X(C_i)$ that contains tuples such that $\tau[X(C_i)] \in C_i$ for all $1 \leq i \leq k$.

Example

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

$\pi_{x_1, x_2}(C_1) \equiv (x_1 \neq x_2)$
$C_1 \cap C_2 \equiv (x_1 < x_2 < x_3)$

Notation and Terminology

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

- 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 to $\mathcal{P}$ is an instantiation I on $X(\mathcal{C})$ which is locally consistent
- I on Y is globally consistent if it can be extended to a solution, i.e., there exists $s \in sol(\mathcal{P})$ with $I = s[Y]$

Example

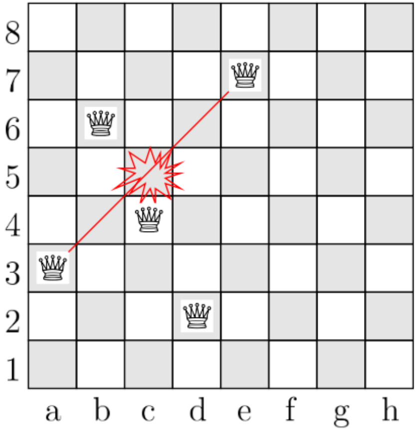
$$\mathcal{P} = \langle X = (x_1, x_2, x_3, x_4), \mathcal{DE} = \{D(x_i) = \{1..5\}, \forall i\},$$
$$\mathcal{C} = \{C_1 \equiv \texttt{alldiff}(x_1, x_2, x_3), C_2 \equiv x_1 \leq x_2 \leq x_3, C_3 \equiv x_4 \geq 2x_2\}\rangle$$

$\pi_{x_1,x_2}(C_1) \equiv (x_1 \neq x_2)$
$I_1 = ((x_1, 1), (x_2, 2), (x_4, 7))$ is not valid
$I_2 = ((x_1, 1), (x_2, 1), (x_4, 3))$ is local consistent since C_3 only one with $X(C_3) \subseteq Y$ 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

- An instantiation I on $\mathcal{P}$ is globally inconsistent if it cannot be extended to a solution of $\mathcal{P}$, globally consistent otherwise.
- A globally inconsistent instantiation is also called a (standard) nogood.
- Remark: A locally inconsistent instantiation is a nogood. The reverse is not necessarily true

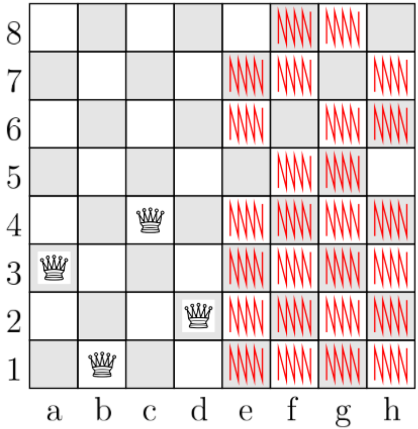
Example



$\{(x_a,3),(x_b,6),(x_c,4),(x_d,2),(x_e,7)\}$ is locally inconsistent

☞ this is a nogood.

Example



$\{(x_a,3),(x_b,1),(x_c,4),(x_d,2)\}$ is globally inconsistent

☞ this is a nogood.

Notation and Terminology

CSP solved by extending partial instantiations to global consistent ones and backtracking at local inconsitencies $\rightsquigarrow$ is NP-complete!

Idea: make the problem more explicit (tighter)

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

Example

$$\begin{aligned}\mathcal{P} =& \langle X = (x_1, x_2, x_3), \mathcal{DE} = \{D(x_i) = [1..4], \forall i\},\\ & \mathcal{C} = \{C_1 \equiv x_1 < x_2, C_2 \equiv x_2 < x_3,\\ & C_3 \equiv \{(111), (123), (222), (333), (234)\}\}\rangle\\ \mathcal{P}' =& \langle X, \mathcal{DE}, \mathcal{C}'\rangle, \mathcal{C}' = \{C_1, C_2, C_3' \equiv \{(123)\}\}\rangle\end{aligned}$$

$\mathcal{P}'$ is a tightening of $\mathcal{P}$: $X_{\mathcal{P}'} = X_{\mathcal{P}}, \quad \mathcal{DE}_{\mathcal{P}'} = \mathcal{DE}$ and $C_1 = C_1', C_2 = C_2', X(C_3) = X(C_3'), C_3' \subset C_3$. All locally inconsitent instantiations on $Y \subseteq X_{\mathcal{P}}$ for $\mathcal{P}$ are locally inconsistent for $\mathcal{P}'$. However not all solutions are preserved.

Example

$$\mathcal{P} = \langle X = (x_1, x_2, x_3), \mathcal{DE} = \{D(x_i) = [1..4], \forall i\},$$
$$\mathcal{C} = \{C_1 \equiv x_1 < x_2, C_2 \equiv x_2 < x_3, C_3 \equiv \{(111), (123), (222), (333)\}\}\rangle$$

$$\mathcal{P}' = \langle X, \mathcal{DE}, \mathcal{C}'\rangle, \mathcal{C}' = \{C_1, C_2, C_3' \equiv \{(123), (231), (312)\}\}\rangle$$

For any tuple τ on $X(C)$ that does not satisfy C there exists a constraint C' in $\mathcal{C}'$ with $X(C') \subseteq X(C)$ such that $\tau[X(C')] \notin C'$ (τ local inconsistent).
Hence $\mathcal{P}' \preceq \mathcal{P}$. But also $\mathcal{P} \preceq \mathcal{P}'$.
$\mathcal{P}'$ is not a tightening of $\mathcal{P}$: $C_3' \not\subseteq$ of any $C \in \mathcal{C}$
They are no-good equivalent.

Notation and Terminology

$\mathcal{S}_{\mathcal{P}}$ 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 $\mathcal{S}^{\mathrm{sol}}_{\mathcal{P}}$ and among them the smallest

$\mathcal{P}^* \in \mathcal{S}^{\mathrm{sol}}_{\mathcal{P}}$ 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 $\rightsquigarrow$ constraint propagation

- Define a property Φ that states necessary conditions on instantiations that enter in the definition of local consistency
- Reduction rules: sufficient conditions to rule out values (or instantiations) that will not be part of a solution (defined through a consistency property φ)
  Rules iteration: set of reduction rules for each constraint that tighten the problem

Constraint Propagation

In general, 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 $\mathcal{C}$

Focus on domain-based tightenings

Domain-based tightenings

The space $\mathcal{S}_{\mathcal{P}}$ of domain-based tightenings of $\mathcal{P}$ is the set of problems $\mathcal{P}' = \langle X', \mathcal{DE}', \mathcal{C}' \rangle$ such that $X_{\mathcal{P}'} = X_{\mathcal{P}}, \quad \mathcal{DE}_{\mathcal{P}'} \subseteq \mathcal{DE}, \quad \mathcal{C}' = \mathcal{C}$

Task:
Finding a tightening $\mathcal{P}^*$ in $\mathcal{S}_{\mathcal{P}}^{\mathrm{sol}} \subseteq \mathcal{S}_{\mathcal{P}}$ (the set that contains all problems that preserve the solutions of $\mathcal{P}$) 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:

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

  (the rule is parameterised by a variable x_i and a constraint C)
  Rules iteration (for all i)

It is clearly NP-hard since it corresponds to solving $\mathcal{P}$ itself.
$\rightsquigarrow$ hence polynomial reduction rules to approximate $\mathcal{P}^*$

Apply rules iteration for each constraint. Domain-based reduction rules are also called propagators.

Example

$C = (|x_1 - x_2| = k)$
Propagator: $D(x_1) \leftarrow D(x_1) \cap [\min_D(x_2) - k .. \max_D(x_2) + k]$

Rather than defining rules we define Φ: e.g., unary, arc, path, k-consistency

Domain-based local consistency

Domain-based local consistency property Φ specifies a necessary condition on values to belong to solutions. We restrict to those stable under union.

A domain-based property Φ is stable under union iff for any Φ-consistent problem $\mathcal{P}_1 = (X, \mathcal{DE}, \mathcal{C})$ and $\mathcal{P}_2 = (X, \mathcal{DE}, \mathcal{C})$ the problem $\mathcal{P}' = (X, \mathcal{DE}_1 \cup \mathcal{DE}_2, C)$ is Φ-consistent.

Example

Φ for each constraint C and variable $x_i \in X(C)$, at least half of the values in $D(x_i)$ belong to a valid tuple satisfying C.

$$\mathcal{P} = \langle X = (x_1, x_2), \mathcal{DE} = \{D_1(x_1) = \{1,2\}, D_1(x_2) = \{2\}\}, C \equiv \{x_1 = x_2\}\rangle$$
$$\mathcal{P} = \langle X = (x_1, x_2), \mathcal{DE} = \{D_2(x_1) = \{2,3\}, D_2(x_2) = \{2\}\}, C \equiv \{x_1 = x_2\}\rangle$$

Both are Φ consistent but they are not stable under union.

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

$\mathrm{sol}(\phi(\mathcal{P})) = \mathrm{sol}(\mathcal{P})$

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

Definition

A set has closure under an operation if performance of that operation on members of the set always produces a member of the same set.

A set is said to be closed under a collection of operations if it is closed under each of the operations individually.

Domain-based tightenings

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 $\mathcal{P}$.

Contrary to $\mathcal{P}^*$, $\Phi(\mathcal{P})$ can be computed by a greedy algorithm:

Corollary If a domain-based consistency property Φ is polynomial to check, finding $\Phi(\mathcal{P})$ is polynomial as well.

enforcing Φ consistency $\equiv$ finding closure $\Phi(\mathcal{P})$

Strength Order

Possible to define a partial order

(For a, b, elements of a poset P, if $a \leq b$ or $b \leq a$, then a and b are comparable. Otherwise they are incomparable)

That is, Φ_1 is at least as strong as another Φ_2 if for any $\mathcal{P}$: $\Phi_1(\mathcal{P}) \leq \Phi_2(\mathcal{P})$,
ie, $X_{\Phi_1(\mathcal{P})} = X_{\Phi_2(\mathcal{P})}, \quad \mathcal{DE}_{\Phi_1(\mathcal{P})} \subseteq \mathcal{DE}_{\Phi_2(\mathcal{P})}, \quad \mathcal{C}_{\Phi_1(\mathcal{P})} = \mathcal{C}_{\Phi_2(\mathcal{P})}$
(any instantiation I on $Y \subseteq X_{\Phi_2(\mathcal{P})}$ locally inconsistent in $\Phi_2(\mathcal{P})$ is locally inconsistent in $\Phi_1(\mathcal{P})$)

Outline

1. Definitions

2. Local Consistency

Node Consistency

We call a CSP node consistent if for every variable x every unary constraint on x coincides with the domain of x.

Example

- $\langle \mathcal{C}, x_1 \geq 0, \ldots, x_n \geq 0; x_1 \in \mathbb{N}, \ldots, x_n \in \mathbb{N} \rangle$
  and $\mathcal{C}$ does not contain unary constraints
  node consistent
- $\langle \mathcal{C}, x_1 \geq 0, \ldots, x_n \geq 0; x_1 \in \mathbb{N}, \ldots, x_n \in \mathbb{Z} \rangle$
  and $\mathcal{C}$ does not contain unary constraints
  not node consistent

A CSP is node consistent iff it is closed under the applications of the Node Consistency rule (propagator):

$$\frac{\langle C; x \in D \rangle}{\langle C; x \in C \cap D \rangle}$$

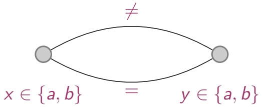
(the rule is parameterised by a variable x and a unary constraint C)

Arc Consistency

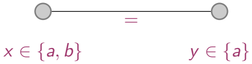
Arc consistency: every value in a domain is consistent with every binary constraint.

- $C = c(x, y)$ with $\mathcal{DE} = \{D(x), D(y)\}$ is arc consistent iff
  - $\forall a \in D(x)$ there exists $b \in D(y)$ such that $(a, b) \in C$
  - $\forall b \in D(y)$ there exists $a \in D(x)$ such that $(a, b) \in C$
- $\mathcal{P}$ is arc consistent iff it is AC for all its binary constraints

In general arc consistency does not imply global consistency.
An arc consistent but inconsistent CSP:



A consistent but not arc consistent CSP:



Generalized Arc Consistency (GAC)

Given arbitrary (non-normalized, non-binary) $\mathcal{P}$, $C \in \mathcal{C}$, $x_i \in X(C)$

(Value) $v \in D(x_i)$ 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)

(Variable) $\mathcal{DE}$ is GAC on C for x_i iff all values in $D(x_i)$ are consistent with C in $\mathcal{DE}$ (i.e., $D(x_i) \subseteq \pi_{\{x_i\}}(C \cap \pi_{\{X(C)\}}(\mathcal{DE}))$)

(Problem) $\mathcal{P}$ 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.

(aka, hyperarc consistency, domain consistency)
Example: arc consistency $\neq$ 2-consistency, AC $<$ 2C on non-normalized binary CSP, and incomparable on arbitrary CSP

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.