DM826 – Spring 2012

Modeling and Solving Constrained Optimization Problems

Lecture 7

Further notions of local consistency

Marco Chiarandini

Department of Mathematics & Computer Science
University of Southern Denmark

Outline

1. Higher Order Consistencies

2. Weaker arc consistencies

3. Generic Rules Iteration

Higher Order Consistencies

- arc consitency does not remove all inconsistencies: even if a CSP is arc consistent there might be no solution
- stronger consistencies techniques are studied:
  - path consistency
  - restricted path consistency
  - k-consistency
  - (i, j)-consistent

Path consistency

Given $\mathcal{P} = \langle X, \mathcal{DE}, \mathcal{C} \rangle$ normalized and x_i, x_j:

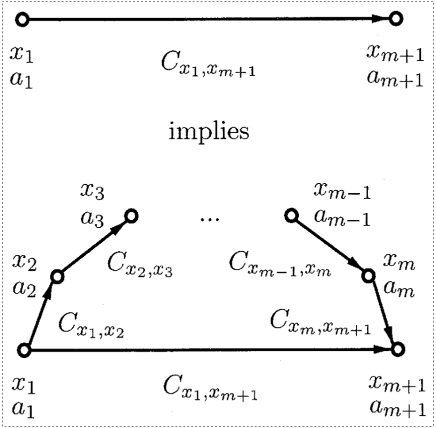
- the pair $(v_i, v_j) \in D(x_i) \times D(x_j)$ is p-path consistent iff forall $Y = (x_i = x_{k_1}, \ldots, x_{k_p} = x_j)$ with $C_{k_q, k_{q+1}} \in \mathcal{C}$ $\exists \tau : \tau[Y] = (v_i = v_{k_1}, \ldots, v_{k_{q+1}} = v_j) \in \pi_Y(\mathcal{DE})$ and $(v_{k_q}, v_{k_{q+1}}) \in C_{k_p, k_{q+1}}$, $q = 1, \ldots, p$
- the CSP $\mathcal{P}$ is p-path consistent iff for any $(x_i, x_j), i \neq j$ any locally consistent pair of values is path consistent.

Example

$$\mathcal{P} = \langle X = (x_1, x_2, x_3), D(x_i) = \{1, 2\}, \mathcal{C} \equiv \{x_1 \neq x_2, x_2 \neq x_3\} \rangle$$

Not path consistent: e.g., for $(x_1, 1), (x_3, 2)$ there is no x_2
$\mathcal{P} = \langle X, \mathcal{DE}, \mathcal{C} \cup \{x_1 = x_3\} \rangle$ is path consistent

Alternative definition:
constraint composition:
$C_{x_1,x_3} = C_{x_1,x_2} \cdot C_{x_2,x_3} = \{(a,b) \mid \exists c((a,c) \in C_{x_1,x_2}, (c,b) \in C_{x_2,x_3})\}$



Example

$$\langle x < y, y < z, x < z; x \in [0..4], y \in [1..5], z \in [6..10] \rangle$$

is path consistent. Indeed:

$$C_{x,z} = \{(a, c) \mid a < c, a \in [0..4], c \in [6..10]\}$$
$$C_{x,y} = \{(a, b) \mid a < b, a \in [0..4], b \in [1..5]\}$$
$$C_{y,z} = \{(b, c) \mid b < c, b \in [1..5], c \in [6..10]\}$$

Example

$$\langle x < y, y < z, x < z; x \in [0..4], y \in [1..5], z \in [5..10] \rangle$$

is not path consistent. Indeed:
$C_{x,z} = \{(a, c) \mid a < c, a \in [0..4], c \in [5..10]\}$ and for $4 \in [0..4]$ and $5 \in [5..10]$ no $b \in [1..5]$ such that $4 < b$ and $b < 5$.

2-path consistency if the path has length 2

- CSP is p-path consistent $\iff$ 2-path consistent (Montanari, 1974). Proof by induction.
- Hence, sufficient to enforce consistency on paths of length 2.
- path consistency algorithms work with path of length two only and, like AC algorithms, make these paths consistent with revisions.
- Even if PC eliminates more inconsitencies than AC, seldom used in practice because of efficiency issues
- PC require extensional representation of constraints and hence huge amount memory.
- Restricted PC does AC and PC only when a variable is left with one value.

k-consistency

Given $\mathcal{P} = \langle X, \mathcal{DE}, \mathcal{C} \rangle$, and set of variables $Y \subseteq X$ with $|Y| = k - 1$:

- a locally consistent instantiation I on Y is k-consistent iff for any kth variable $x_{i_k} \in X \setminus Y$ $\exists$ a value $v_{i_k} \in D(x_{i_k}) : I \cup \{x_{i_k}, v_{i_k}\}$ is locally consistent
- the CSP $\mathcal{P}$ is k-consistent iff for all Y of $k - 1$ variables any locally consistent I on Y is k-consistent.

Example

arc-consistent $\neq$ 2-consistent

$$D(x_1) = D(x_2) = \{1, 2, 3\}, x_1 \leq x_2, x_1 \neq x_2$$

arc consistent, every value has a support on one constraint
not 2-consistent, $x_1 = 3$ cannot be extended to x_2 and $x_2 = 1$ not to x_1 with both constraints

Example

$$D(x_i) = \{1, 2\}, i = 1, 2, 3; \mathcal{C} = \{(1, 1, 1), (2, 2, 2)\}$$

is $\mathcal{P}$ path consistent? Yes because no binary variable such that $X(C) \subseteq Y$
is $\mathcal{P}$ 3-consistent? No, because $(x_1, 1), (x_2, 2)$ is locally consistent but cannot be extended consistently to x_3.

Example

- $\langle x_1 \neq x_2, x_1 \neq x_3, x_1 \neq x_3; x_1 \in \{0, 1\}, x_2 \in \{0, 1\}, x_3 \in \{0, 1\}\rangle$
- $\langle x_1 \neq x_2, x_1 \neq x_3; x_1 \in \{a, b\}, x_2 \in \{a\}, ..., x_k \in \{a\}\rangle$

Given $k > 1$.

- there exists a CSP that is $(k-1)$-consistent but not k-consistent
- there exists a CSP that is not $(k-1)$-consistent but is k-consistent

- $\mathcal{P}$ is strongly k-consistent iff it is j-consistent $\forall j \leq k$
- constructing one requires $O(n^k d^k)$ time and $O(n^{k-1} d^{k-1})$ space.
- if $\mathcal{P}$ is strongly n-consistent then it is globally consistent
- (i, j)-consistent: any consistnt instantiation of i different variables can be extended to a consistent instantiation including any j additional variables
  k consistency $\equiv (k-1, k)$ consistent
- strongly (i, j)-consistent

Outline

1. Higher Order Consistencies
2. Weaker arc consistencies
3. Generic Rules Iteration

Weaker arc consistencies

- reduce calls to Revise in coarse-grained algorithms (Forward Checking)
- reduce amount of work of Revise (Bound consistency)

Forward checking

Given $\mathcal{P}$ binary and $Y \subseteq X \ : |D(x_i)| = 1 \forall x_i \in Y$:

- $\mathcal{P}$ is forward checking consistent according to instantiation I on Y iff it is locally consistent and for all $x_i \in Y$, for all $x_j \in X \setminus Y$ for all $C(x_i, x_j) \in \mathcal{C}$ is arc consistent on $C(x_i, x_j)$.

  (all constraints between assigned and not assigned variables are consistent.)

- $O(ed)$ time
- Extension to non-binary constraints
- A search procedure maintaining forward checking does not need to check consistency of values of the current variable against already instantiated ones $\neq$ chronological backtracking

Lookahead

Defined only by procedure, not by fixed point definition

Algorithm partial lookahead and full lookahead

procedure $PL(N, Y, x_i)$;
$FC(N, Y, x_i)$;
foreach $j \leftarrow i + 1$ ***to*** n **do**
    foreach $k \leftarrow j + 1$ ***to*** $n \mid c_{jk} \in C_N$ **do**
        if ***not*** *Revise*(x_j, c_{jk}) **then** return false

procedure $FL(N, Y, x_i)$;
$FC(N, Y, x_i)$;
foreach $j \leftarrow i + 1$ ***to*** n **do**
    foreach $k \leftarrow i + 1$ ***to*** $n, k \neq j \mid c_{jk} \in C_N$ **do**
        if ***not*** *Revise*(x_j, c_{jk}) **then** return false

Bound consistency

- domains inherit total ordering on $\mathbf{Z}$, $\min_D(x)$ and $\max_D(x)$ called bounds of $D(x)$
- Given $\mathcal{P}$ and C, a bounded support τ on C is a tuple that satisfies C and such that for all $x_i \in X(C)$, $\min_D(x_i) \leq \tau[x_i] \leq \max_D(x_i)$, that is, $\tau \in C \cup \pi_{X(C)}(D^I)$ (instead of D)

$$D^I(x_i) = \{v \in \mathbf{Z} \mid \min_D(x_i) \leq v \leq \max_D(x_i)\}$$

- C is bound($\mathbf{Z}$) consistent iff $\forall x_i \in X$ its bounds belong to a bounded support on C
- C is range consistent iff $\forall x_i \in X$ all its values belong to a bounded support on C
- C is bound($\mathbf{D}$) consistent iff $\forall x_i \in X$ its bounds belong to a support on C

- $GAC < (\mathrm{bound}(\mathbf{D}), \mathrm{range}) < \mathrm{bound}(\mathbf{Z})$ (strictly stronger)
  bound(**D**) and range are incomparable
- most of the time gain in efficiency

Example

$$\mathrm{sum}(x_1, \ldots, x_k, k)$$

GAC is NP-complete (reduction from SubSet problem).
But bound(**Z**) is polynomial: test $\forall 1 \leq i \leq n$:
$\min_D(x_i) \geq k - \sum_{j \neq i} \max_D(x_j)$
$\max_D(x_i) \leq k - \sum_{j \neq i} \min_D(x_j)$

Outline

1. Higher Order Consistencies

2. Weaker arc consistencies

3. Generic Rules Iteration

Algorithms for constraint propagation:

- scheduling steps of atomic reduction
- termination criterion: local consistency

- How to schedule the application of reduction rules to guarantee termination?
- How to avoid (at low cost) the application of redunant rules?
- Have all derivations the same result?
- How can we characterize it?

Propagators

- Given $\mathcal{P}$ a reduction rule is a function f from $\mathcal{S}_\mathcal{P}$ to $\mathcal{S}_\mathcal{P}$ for all $\mathcal{P}' \in \mathcal{S}_\mathcal{P}$, $f(\mathcal{P}') \in \mathcal{S}_\mathcal{P}$.
  (most cases take care of one a single variable and a single constraints):
- Given $\mathcal{C}$ in $\mathcal{P}$ a propagator f for $\mathcal{C}$ is a reduction rule from $\mathcal{S}_\mathcal{P}$ to $\mathcal{S}_\mathcal{P}$ that tightens only domains independently of the constraints other than C.
- Properties of propagators:
  Given $\mathcal{P}$, f can be:
  - contracting: $f(\mathcal{P}) \leq \mathcal{P}$
  - monotonic if $\mathcal{P}_1 \leq \mathcal{P}_2 \Rightarrow f(\mathcal{P}_1) \leq f(\mathcal{P}_2)$
  - idempotent if $f(f(\mathcal{P})) = f(\mathcal{P})$
  - commuting if $fg(\mathcal{P}) = gf(\mathcal{P})$
  - subsumed by $\mathcal{P}$ iff $\forall \mathcal{P}_1 \leq \mathcal{P} : f(\mathcal{P}_1) = \mathcal{P}_1$

- Iteration: Let $\mathcal{P} = \langle X, \mathcal{DE}, \mathcal{C} \rangle$ and $F = \{f_1, \ldots, f_k\}$ a finite set of propagators on $\mathcal{S}_{\mathcal{P}}$. An iteration of F on $\mathcal{P}$ is a sequence $\langle \mathcal{P}_0, \mathcal{P}_1, \ldots \rangle$ of elements of $\mathcal{S}_{\mathcal{P}}$ defined by
$$\mathcal{P}_0 = \mathcal{P}$$
$$\mathcal{P}_j = f_{n_j}(\mathcal{P}_{j-1})$$
where $j > 0$ and $n_j \in [1, \ldots, k]$.
- $\mathcal{P}$ is stable for F iff $\forall f \in F, f(\mathcal{P}) = \mathcal{P}$
- there may be several stable $\mathcal{P}$ but if F are monotonic then unique
- Let $\mathcal{P} = \langle X, \mathcal{DE}, \mathcal{C} \rangle$ and $F = \{f_1, \ldots, f_k\}$. If $\langle \mathcal{P}_0, \mathcal{P}_1, \ldots \rangle$is infinite iteration of F where each $f \in F$ is activated infinitely often then there exists $j \geq 0$ such that $\mathcal{P}_j$ is stable for F (j is finite)
- If $\mathcal{P}$ is stable for F then it is its weakest simultaneous fixed point (greatest mutual fixed point of all propagators).
A strongest simultaneous fixed point would be a solution (hence not unique) which would violate solution preservation

Iteration of Reduction Rules

procedure *Generic-Iteration*(N, F);
  $G \leftarrow F$;
  while $G \neq \emptyset$ **do**
    select and remove g from G;
    if $N \neq g(N)$ **then**
      $update(G)$;
      $N \leftarrow g(N)$;

  /* $update(G)$ adds to G at least all functions f in $F \setminus G$ for which
  $g(N) \neq f(g(N))$ */

If the propagator is contracting then Generic-Iteration terminates.
If propagator is monotonic then the final result does not change with the order in which propagators are applied.

Example

$$\forall N_1 = (X, D_1, C) \in \mathcal{P}_{ND}, \forall x_i \in X, \forall c_j \in C,\ f_{i,j}(N_1) = (X, D_1', C) \text{ with}$$

$$D_1'(x_i) = \pi_{\{x_i\}}(c_j \cap \pi_{X(c_j)}(D_1)) \text{ and } D_1'(x_k) = D_1(x_k), \forall k \neq i.$$

Set of propagators $F_{AC} = \{f_{ij} \mid x_i \in X, c_j \in \mathcal{C}\}$ all monotonic. $\Rightarrow$ terminates in arc consistency closure, which is fixed point for F_{AC}.

References

Apt K.R. (2003). **Principles of Constraint Programming**. Cambridge University Press.

Barták R. (2001). **Theory and practice of constraint propagation**. In *Proceedings of CPDC2001 Workshop*, pp. 7–14. Gliwice.

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.