Soft Constraints Processing over Divisible Residuated Lattices

Simone Bova
bova@dico.unimi.it

Department of Computer Science
University of Milan (Milan, Italy)

ECSQARU 2009
1-3 July 2009, Verona (Italy)

Outline

Outline

Constraint Satisfaction Problems

Problem: CSP

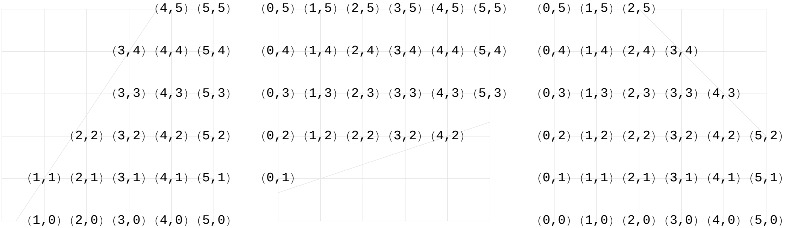
Instance: (X, D, P) where:

- *(i)* X is a finite set of *variables*;
- *(ii)* D is a finite set of *values* (aka *domain*);
- *(iii)* $P = \{C_1, \ldots, C_q\}$ is a finite set of *constraints*, that is, pairs $(\mathbf{x}_i, R_i)$ having $\mathbf{x}_i \in X^m$ as *scope* and $R_i \subseteq D^m$ as *relation*.

Question: Is there an *assignment* $f : X \rightarrow D$ *satisfying* all constraints, that is, such that $f(\mathbf{x}_i) \in R_i$ for all $i \in \{1, \ldots, q\}$?

CSP | Example

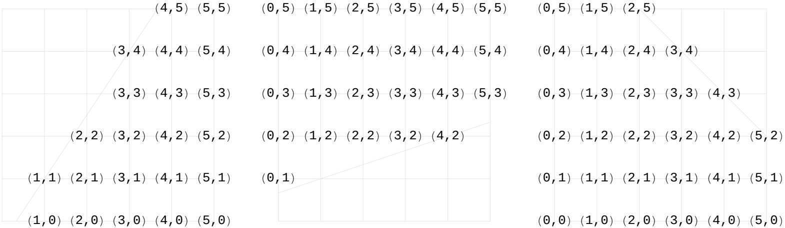
$\{R_1(x_1, x_2), R_2(x_1, x_2), R_3(x_1, x_2)\}$ with $R_1, R_2, R_3 \subseteq \{0, \ldots, 5\}^2$:



(a) R_1. (b) R_2. (c) R_3.

CSP | Example

$\{R_1(x_1, x_2), R_2(x_1, x_2), R_3(x_1, x_2)\}$ with $R_1, R_2, R_3 \subseteq \{0, \dots, 5\}^2$:

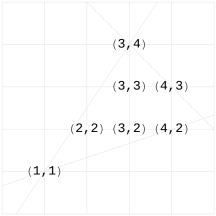


(a) R_1. (b) R_2. (c) R_3.

Is there $f\colon \{x_1, x_2\} \to \{0, \dots, 5\}$ satisfying all constraints?

CSP | Example

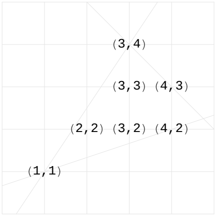
There are several such f's…



(a) $R_1 \cap R_2 \cap R_3$.

CSP | Example

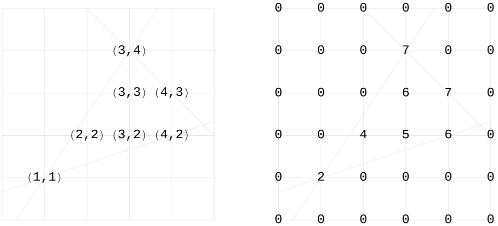
There are several such f's... what if they pay $f(x_1) + f(x_2)$ euro?



(a) $R_1 \cap R_2 \cap R_3$.

CSP | Example

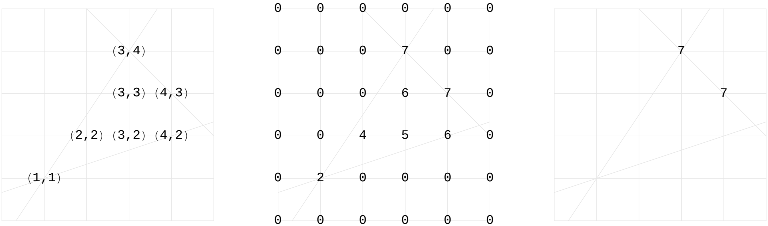
There are several such f's. . . what if they pay $f(x_1) + f(x_2)$ euro?



(a) $R_1 \cap R_2 \cap R_3$. (b) f' venue.

CSP | Example

There are several such f's. . . what if they pay $f(x_1) + f(x_2)$ euro?



(a) $R_1 \cap R_2 \cap R_3$. (b) f' venue. (c) Optimal f's.

Feasibility vs. Optimization

The *crisp* CSP is a *feasibility* problem
(any satisfying assignment is equally good).

The *soft* CSP is an *optimization* problem: each constraint *maps* assignments to a *valuation structure*, that is, a bounded poset equipped with a suitable *combination* operator; the goal is to find an assignment such that the combination of its images under all the constraints is *maximal* in the structure.

Valuation Structure | Example (Cont'd)

Step 1: Design valuation structure.

$\mathbf{A} = (\{0, \ldots, 10\}, \bot = 0 < \cdots < 10 = \top, \mathit{min})$. min:

(*i*) associative, commutative (no precedence, no order);

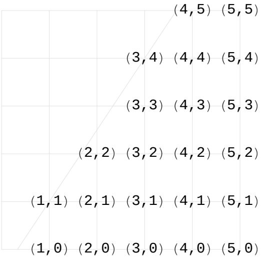
(*ii*) monotone over $\leq$ (more constraints, worst solutions);

(*iii*) $\mathit{min}\{x, \bot\} = \bot$ (unsatisfiability marker);

(*iv*) $\mathit{min}\{x, \top\} = x$ (triviality marker).

Soft Constraints | Example (Cont'd)

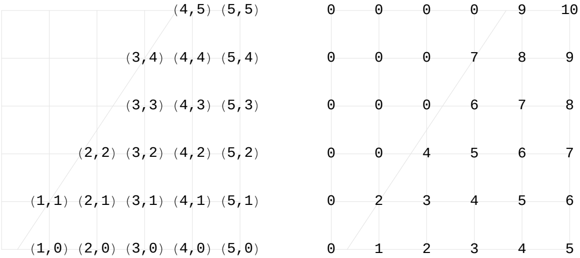
Step 2: Soften crisp constraints (map assignments to the structure).



(a) Crisp R_1.

Soft Constraints | Example (Cont'd)

Step 2: Soften crisp constraints (map assignments to the structure).

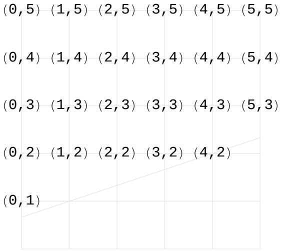


(a) Crisp R_1. (b) Soft R_1.

Figure: $R_1 : \{0, \ldots, 5\}^2 \rightarrow \{0, \ldots, 10\}$.

Soft Constraints | Example (Cont'd)

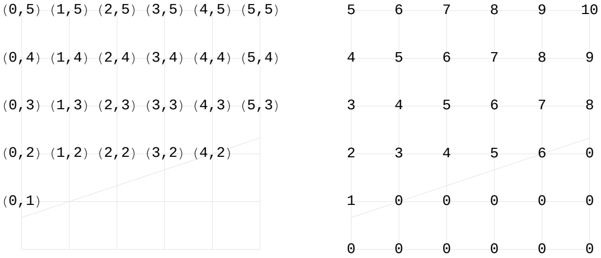
Step 2: Soften crisp constraints (map assignments to the structure).



(a) Crisp R_2.

Soft Constraints | Example (Cont'd)

Step 2: Soften crisp constraints (map assignments to the structure).

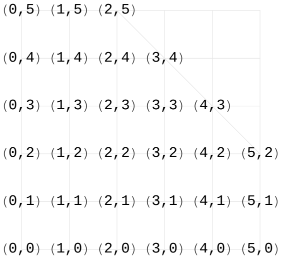


(a) Crisp R_2. (b) Soft R_2 image.

Figure: $R_2 \colon \{0, \dots, 5\}^2 \to \{0, \dots, 10\}$.

Soft Constraints | Example (Cont'd)

Step 2: Soften crisp constraints (map assignments to the structure).



(a) Crisp R_3.

Soft Constraints | Example (Cont'd)

Step 2: Soften crisp constraints (map assignments to the structure).

(0,5)	(1,5)	(2,5)			
(0,4)	(1,4)	(2,4)	(3,4)		
(0,3)	(1,3)	(2,3)	(3,3)	(4,3)	
(0,2)	(1,2)	(2,2)	(3,2)	(4,2)	(5,2)
(0,1)	(1,1)	(2,1)	(3,1)	(4,1)	(5,1)
(0,0)	(1,0)	(2,0)	(3,0)	(4,0)	(5,0)

(a) Crisp R_3.

5	6	7	0	0	0
4	5	6	7	0	0
3	4	5	6	7	0
2	3	4	5	6	7
1	2	3	4	5	6
1	1	2	3	4	5

(b) Soft R_3 image.

Figure: $R_3 : \{0, \ldots, 5\}^2 \to \{0, \ldots, 10\}$.

Combination and Maximization | Example (Cont'd)

Step 3: Maximize constraints combination. For instance,

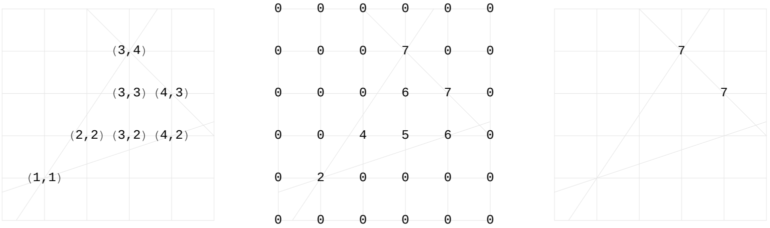
$$(2,4) \;\Rightarrow min\{R_1(2,4), R_2(2,4), R_3(2,4)\} = min\{0,6,6\} = 0,$$
$$(3,2) \;\Rightarrow min\{R_1(3,2), R_2(3,2), R_3(3,2)\} = min\{5,5,5\} = 5, \ldots$$

Combination and Maximization | Example (Cont'd)

Step 3: Maximize constraints combination. For instance,

$(2,4) \;\Rightarrow min\{R_1(2,4), R_2(2,4), R_3(2,4)\} = min\{0,6,6\} = 0,$

$(3,2) \;\Rightarrow min\{R_1(3,2), R_2(3,2), R_3(3,2)\} = min\{5,5,5\} = 5, \ldots$



(a) Crisp solutions. (b) Soft solutions. (c) Optimal solutions.

Definition

Definition (Soft CSP)

A *soft CSP* is a tuple $\mathbf{P} = (X, D, P, \mathbf{A})$ with:

(*i*) *variables* $X = \{1, \dots, n\} = [n]$;

(*ii*) finite *domains* $D = (D_i)_{i \in [n]}$ where i ranges over D_i;

(*iii*) *valuation structure* $\mathbf{A} = (A, \leq, \odot, \top, \bot)$ st $(A, \leq, \top, \bot)$ is a bounded poset, $(A, \odot, \top)$ is a commutative monoid, $\odot$ is monotone over $\leq$ (that is, $x \leq y$ implies $z \odot x \leq z \odot y$);

(*iv*) P finite multiset of *constraints* of the form

$$C_Y : \prod_{i \in Y} D_i \to A,$$

where $Y \subseteq X$ is the *scope* of C_Y.

Definition

Notation ($Y \subseteq X$): $l(Y) = \prod_{i \in Y} D_i$; $t|_Y$ projects $t \in l(X)$ onto Y.

Definition (Solution, Inconsistence, Equivalence)

Any $t \in l(X)$ such that $\bigodot_{C_Y \in P} C_Y(t|_Y)$ is maximal wrt $\leq$ in

$$S(\mathbf{P}) = \{ \bigodot_{C_Y \in P} C_Y(t|_Y) \mid t \in l(X)\} \subseteq A$$

is a *solution* to $\mathbf{P}$, and $\mathbf{P}$ is *inconsistent* if $S(\mathbf{P}) = \{\bot\}$.
$\mathbf{P} = (X, D, P, \mathbf{A})$ is *equivalent* to $\mathbf{P}' = (X, D, P', \mathbf{A})$
iff for every $t \in l(X)$,

$$\bigodot_{C_Y \in P} C_Y(t|_Y) = \bigodot_{C_Y \in P'} C_Y(t|_Y).$$

Logical Structures

Fact

A CSP is a soft CSP $(X, D, P, \mathbf{A})$ *where:*

(i) $D = (D_i)_{i \in X}$ *with* $|\{D_i \mid i \in X\}| = 1$*;*

(ii) $\mathbf{A} = (\{0, 1\}, 0 < 1, min, 1, 0)$.

In the crisp CSP, **A** is a reduct of the Boolean algebra **2**, the algebraic counterpart of classical logic.

Logical Structures

Fact

A CSP is a soft CSP $(X, D, P, \mathbf{A})$ *where:*

(i) $D = (D_i)_{i \in X}$ *with* $|\{D_i \mid i \in X\}| = 1$*;*

(ii) $\mathbf{A} = (\{0, 1\}, 0 < 1, min, 1, 0)$.

In the crisp CSP, $\mathbf{A}$ is a reduct of the Boolean algebra **2**, the algebraic counterpart of classical logic.

Proposal: Adopt algebraic counterparts of nonclassical logics as valuation structures for the soft CSP.

Residuated Lattices

In Boolean logic the relation between *conjunction*, $\wedge$, and *implication*, $\rightarrow$, is given by the *residuation* equivalences,

$$x \wedge y \leq z \text{ iff } x \leq y \rightarrow z \text{ iff } y \leq x \rightarrow z,$$

which imply many of the properties of $\wedge$ and $\rightarrow$ (commutativity of $\wedge$, distributivity of $\wedge$ over $\vee$, left-distributivity of $\rightarrow$ over $\vee$, and right-distributivity of $\rightarrow$ over $\wedge$).

The prominent approach in generalizing Boolean logic relies upon generalizing Boolean conjunction, by means of a binary operation, $\odot$, called *fusion*, and imposing the residuation equivalences with $\wedge$ replaced by $\odot$.

Residuated Lattices

Definition (Commutative Bounded Residuated Lattice, CBRL)

A *(commutative bounded) residuated lattice* is an algebra $(A, \vee, \wedge, \odot, \rightarrow, \top, \bot)$ of type $(2, 2, 2, 2, 0, 0)$ st:

(i) $(A, \odot, \top)$ is a commutative monoid;

(ii) $(A, \vee, \wedge, \top, \bot)$ is a bounded lattice;

(iii) *residuation* holds, that is $x \odot y \leq z$ if and only if $y \leq x \rightarrow z$.

The monotonicity of fusion over the order follows.

Lattice Orders and Nonidempotent Combinations

$Y \subseteq X, t, t' \in l(Y)$, **A** *CBRL*.

- $C_Y(t) \leq C_Y(t')$ says that t' is preferred to t (the distance between $C_Y(t)$ and $C_Y(t')$ gives the degree of such preference, ranging over **A**'s *depth*).
- $C_Y(t) \parallel C_Y(t')$ says that t' and t are incomparable (**A**'s *width* gives the number of simultaneous rankings supported by **A**).
- $\wedge$'s and $\vee$'s required by algorithmics (tentative).
- $C_Y(t) \odot C_Y(t) < C_Y(t)$ says that repetitions matter.

Outline

Soft Constraints and Logical Structures
- Soft Constraint Satisfaction Problems
- Commutative Bounded Residuated Lattices

Soft Constraints Processing
- Enforcing Algorithms
- k-Hyperarc Consistency

Conclusion

Soft CSP

Problem: SOFT-CSP

Instance: $(X, D, P, \mathbf{A})$

Goal: Find $t \in l(X)$ maximizing $\bigodot_{C_Y \in P} C_Y(t|_Y)$ in $\mathbf{A}$.

The SOFT-CSP is NP-hard:

(*i*) characterize tractable cases (theoretical side);

(*ii*) leverage exhaustive search (*enforcing* algorithms, applicative side).

Enforcing Algorithms

Given a soft CSP, an *enforcing* algorithm enforces over it a *local consistency* property, in polynomial time.

Either the input problem is found locally (hence, globally) inconsistent, or it is transformed into an *equivalent* problem, possibly inconsistent but *easier* (with a smaller solution space).

Despite their incompleteness as inconsistency test, enforcing algorithms are useful as subprocedures in exhaustive search methods (*branch and bound*).

Divisible Residuated Lattices

What is the additional structure required to implement enforcing algorithms over *CBRL*?

Divisible Residuated Lattices

What is the additional structure required to implement enforcing algorithms over *CBRL*? *Divisibility* is necessary...

Definition (GBL-algebra)

A *GBL-algebra* is a *CBRL* where *divisibility* holds, that is, $x \wedge y = x \odot (x \to y)$.

Divisible Residuated Lattices

What is the additional structure required to implement enforcing algorithms over *CBRL*? *Divisibility* is necessary...

Definition (GBL-algebra)

A *GBL-algebra* is a *CBRL* where *divisibility* holds, that is, $x \wedge y = x \odot (x \to y)$.

GBL-algebras have a natural logical interpretation, the intersection of Basic (fuzzy) logic and intuitionistic logic.

Adopting valuation structures with a logical interpretation, enforcing algorithms reduce to logical deductions (refutations).

k-Hyperarc Consistency

A soft CSP is *k-hyperarc consistent* if it is possible to extend any *consistent* assignment of a variable i to an assignment of any other $\leq k-1$ variables, constrained by i, avoiding additional costs [BG06, CS04, LS04].

Notation ($Y \subseteq X, i \in Y, a \in D_i, t \in l(Y \setminus \{i\})$):
$(t \cdot a) = t' \in l(Y)$ st $t'|_{\{i\}} = a$ and $t'|_{Y \setminus \{i\}} = t$.

Definition (k-Hyperarc Consistency)

$\mathbf{P} = (X, D, P, \mathbf{A})$ soft CSP, $Y \subseteq X$ st $2 \leq |Y| \leq k$ and $C_Y \in P$. Y is *k-hyperarc consistent* if for each $i \in Y$ and each $a \in D_i$ such that $C_{\{i\}}(a) > \bot$, there exists $t \in l(Y \setminus \{i\})$ such that,

$$C_Y(t \cdot a) = \top.$$

P is *k-hyperarc consistent* if every $Y \subseteq X$ st $2 \leq |Y| \leq k$ and $C_Y \in P$ is k-hyperarc consistent.

Specification

Algorithm: k-HYPERARCCONSISTENCY

Input: A soft CSP $\mathbf{P} = (X, D, P, \mathbf{A})$, where $\mathbf{A}$ is *GBL*-algebra.

Output: $\perp$, or a k-hyperarc consistent soft CSP, equivalent to $\mathbf{P}$.

Pseudocode | 1

k-HYPERARCCONSISTENCY($(X, D, P, \mathbf{A})$)

$Q \leftarrow \{1, \ldots, n\}$
while $Q \neq \emptyset$ **do**
  $i \leftarrow$ POP(Q)
  foreach $Y \subseteq X$ such that $2 \leq |Y| \leq k$, $i \in Y$ and $C_Y \in P$ **do**
    domainShrink $\leftarrow$ PROJECT(Y, i)
    if $C_{\{i\}}(a) = \bot$ for each $a \in D_i$ **then**
      return $\bot$
    else if domainShrink **then**
      PUSH(Q, i)
    endif
  endforeach
endwhile
return $(X, D, P', \mathbf{A})$

Pseudocode | 2

PROJECT(Y, i)

domainShrink $\leftarrow$ **false**
foreach $a \in D_i$ such that $C_{\{i\}}(a) > \bot$ **do**
  $x \leftarrow$ a maximal element in $\{C_Y(t \cdot a) \mid t \in l(Y \setminus \{i\})\}$
  $C_{\{i\}}(a) \leftarrow C_{\{i\}}(a) \odot x$
  if $C_{\{i\}}(a) = \bot$ **then**
    domainShrink $\leftarrow$ **true**
  endif
  foreach $t \in l(Y \setminus \{i\})$ **do**
    $C_Y(t \cdot a) \leftarrow (x \rightarrow C_Y(t \cdot a))$
    $\triangleright$ by divisibility, $z \leq x$ implies $(y \odot x) \odot (x \rightarrow z) = y \odot z$
  endforeach
endforeach
return domainShrink

Correctness and Complexity

Lemma (Complexity)

Let $\mathbf{P} = (X, D, P, \mathbf{A})$ *be soft CSP with* $X = [n]$, $d = \max_{i \in [n]} |D_i|$ *and* $e = |P|$. *Then,* k-HYPERARCCONSISTENCY($\mathbf{P}$) *runs in* $O(e^2 \cdot d^{k+1})$ *time.*

Lemma (Soundness)

Let $\mathbf{P} = (X, D, P, \mathbf{A})$ *be a soft CSP. Consider the output of* k-HYPERARCCONSISTENCY($\mathbf{P}$):

(i) *if it is* $\perp$, *then* $\mathbf{P}$ *is inconsistent;*

(ii) *ow it is a k-hyperarc consistent soft CSP equivalent to* $\mathbf{P}$.

Outline

Soft Constraints and Logical Structures

Soft Constraint Satisfaction Problems

Commutative Bounded Residuated Lattices

Soft Constraints Processing

Enforcing Algorithms

k-Hyperarc Consistency

Conclusion

Summary

We presented certain subvarieties of commutative bounded residuated lattices as *natural* valuation structures for soft CSP's.

These structures constitute the algebraic counterparts of a large family of nonclassical logics, and provide a uniform *logical* interpretation of enforcing procedures.

Divisibility supports a sound implementation of standard enforcing procedures.

References

S. Bistarelli and F. Gadducci.
Enhancing Constraints Manipulation in Semiring-Based Formalisms.
ECAI 2006, 63–67, 2006.

M. C. Cooper and T. Schiex.
Arc Consistency for Soft Constraints.
Artificial Intellingence, 154(1-2):199–227, 2004.

N. Galatos, P. Jipsen, T. Kowalski, and H. Ono.
Residuated Lattices: An Algebraic Glimpse at Substructural Logics.
Elsevier, 2007.

J. Larrosa and T. Schiex.
Solving Weighted CSP by Maintaining Arc Consistency.
Artificial Intellingence, 159(1-2):1–26, 2004.

Thanks!