

# Soft constraint processing

Thomas Schiex (INRA, Toulouse, France)

July 27, 2005

# Contents

<b>1</b>	<b>Soft constraint processing</b>	<b>1</b>
1.1	Notations	1
1.2	Soft constraint networks	2
1.2.1	Valued constraint networks	2
1.2.2	Specific instances	4
1.2.3	Operations on cost functions	6
1.2.4	Related frameworks	6
1.2.4.1	Semiring constraint networks	7
1.2.4.2	Constrained optimization	7
1.2.4.3	Others	8
1.3	Search algorithms	8
1.3.1	Branch and bound	8
1.3.2	Russian Doll Search	10
1.3.3	Local search	10
1.3.4	Experimenting with soft CN	12
1.4	Inference algorithms	12
1.4.1	Complete inference	14
1.4.1.1	Variable elimination	14
1.4.2	Incomplete inference	16
1.4.2.1	Mini-buckets	16
1.4.2.2	Soft local consistency	17
1.4.2.3	Dealing with more operators	18
1.4.2.4	Soft global constraints	22
1.4.3	Polynomial classes	22
1.4.3.1	Fuzzy networks	22
1.4.3.2	Weighted CN	23
1.5	Combining search and inference	24
1.5.1	Direct combination	24
1.5.2	Exploiting stronger bounds	25
1.5.2.1	Local consistency based bounds	25
1.5.2.2	Mini-buckets based bounds	25
1.6	Using soft constraints	26
1.6.1	Existing solvers and resources	26
1.6.2	Some applications	26
1.6.3	Frequency assignment	27

### Abstract

This (not so) short text presents some results I'm aware of in the field of soft constraint network processing. Given the short (four hours) format of the school (and the relatively short time I used to prepare the notes) it does not try to give an exhaustive picture of the current state of the field (although many references have been included). It was simply not possible.

After a short introduction with motivations, the first section gives some chosen examples of existing soft constraint frameworks, starting from my favorite generic framework (valued constraint network) and relating it to other more or less general frameworks including constrained optimization.

Then, we introduce fundamental operations on soft constraints (aka cost functions) and show how these fundamental operations can be used to process and solve soft constraint networks by systematic (branch and bound) or local *search*, by complete or incomplete *inference* and finally using hybrid methods. Some polynomial classes for soft constraints are given.

Some existing applications, resources and solvers for soft constraints are finally outlined. Note also that another overview on the area appeared in [P. Meseguer et al., 2003].

Very likely, this text will contain errors for which you can directly blame me (after having convinced me that it is an error). Thanks for telling me.

# Chapter 1

## Soft constraint processing

### Introduction

The classical constraint network framework is essentially oriented towards satisfaction of all constraints. The initial credo of the constraint satisfaction problem is that finding an optimal solution is rarely useful, it suffices to find a satisfactory one. Modeling a problem consists just in identifying decision variables, associated domains and a list of constraints that states the properties that are needed to define a “satisfactory” object. In time tabling for example, this includes collecting physical constraints representing for example the fact that a person cannot be at two different locations at the same time (or a room occupied by two simultaneous courses) but also the fact that some powerful professor prefers to avoid courses on Friday afternoon. Despite the unavoidable difficulties of NP-hard problem solving, this approach has been very successful in practice in several domains. However, in some cases, the problem defined in this way is overconstrained: there is no “satisfactory” solution. In this case, several phases of model refinement may be needed to heuristically relax some constraints. This process, when it is feasible, is extremely time consuming and rarely formalized.

As the time-tabling example shows, overconstrained problems often appear when constraints are used to formalize desired properties rather than real constraints that cannot be violated. Such desired properties are not faithfully represented by constraints but should rather be considered as “soft constraints” whose violation should be avoided as far as possible.

This initial motivation has led to the design of several extensions of the classical constraint network model allowing to state beforehand how constraints should be relaxed in case of overconstrainedness. Actually, it appears that these frameworks are also very adequate to capture constrained optimization problems (where the usual set of constraints is completed with a specific criteria to optimize). In practice, even in consistent pure decision problems, users will easily tell you that they prefer some solutions and soft constraints can often capture such preferences.

When shifting from classical to soft constraint networks, two problems must be solved: how should I modify the classical CN framework to enable the expression of soft constraints, and then how can I tackle the resulting optimization problem.

Even if the first true soft constraint paper (which explicitly extends classical constraint networks) I am aware of was written by [Rosenfeld et al., 1976], the field became really active in the nineties. Since, a lot of successful exploration has been done to the point where a large subset of the classical constraint network toolbox (theorems, properties, polynomial classes, algorithms, solvers) have been extended (or shown not to extend) to cost functions. But there are still large unexplored areas and a lot of work remains to be done.

### 1.1 Notations

The document essentially uses basic mathematical notions and notations. A  $k$ -tuple is a sequence of  $k$  objects denoted  $(v_1, \dots, v_k)$ . The  $i^{th}$  element of a tuple  $t$  is denoted  $t[i]$ . The Cartesian product of sets

$A_1, \dots, A_k$ , denoted as  $A_1 \times \dots \times A_k$  or  $\prod_{i=1}^k A_i$  is the set of all  $k$ -tuples  $(v_1, \dots, v_k)$  such that  $v_i \in A_i$  for all integers  $i \in [1, k]$ .

A variable represents an unknown element of its domain, assumed to be a finite set of values here. Given a sequence of variables  $S = (x_1, \dots, x_k)$  and their domains  $D_1, \dots, D_k$ , a relation  $R$  on  $S$  is a subset of  $D_1 \times \dots \times D_k$  also denoted  $\ell(S)$ . A value  $a$  in  $D_i$  is often denoted  $(i, a)$ . The relation has scope  $S$ , arity  $|S|$ . When needed, we put an explicit emphasis on the scope of a relation  $t_S \in R_S$  which is an assignment of  $S$ . For  $S' \subseteq S$ ,  $t_S[S']$  is used to denote the projection of  $t_S$  on  $S'$  which is the subtuple of  $t$  obtained by removing values for variables not in  $S'$ . Given a tuple  $t$  on  $S$ ,  $x_i \notin S$  and  $a \in D_i$ , the tuple  $t \cdot (i, a)$  is the tuple over  $S \cup \{x_i\}$  such that  $t \cdot (i, a)[S] = t$  and  $t \cdot (i, a)[\{x_i\}] = a$ .

Given a relation  $R_S$  and a set  $S' \subset S$  of variables, projecting  $R_S$  on  $S'$  produces a new relation  $R_{S'} = R_S[S']$  and  $t' \in R_{S'}$  iff  $\exists t \in R_S, t[S'] = t'$ . Given two tuples  $t_S$  and  $t_T$  such that  $t_S[S \cap T] = t_T[S \cap T]$ , their *join*  $t_S \bowtie t_T$  is a tuple  $t_{S \cup T}$  such that  $t_{S \cup T}[S] = t_S$  and  $t_{S \cup T}[T] = t_T$ . Given two relations  $R_S$  and  $R_T$ , their *join*  $R_S \bowtie R_T$  is a new relation  $R_{S \cup T}$  formed by all possible joins of tuples of  $R_S$  with tuples of  $R_T$ .

A classical constraint network (CN)  $(X, D, C)$  is defined by:

- a set of *variables*  $X = \{x_1, \dots, x_n\}$
- a set of *domains*  $D = \{D_1, \dots, D_n\}$
- a set of *e constraints*  $C$ .

A constraint  $c \in C$  is a relation on a sequence of variables  $S$ , denoted  $c_S$ .  $|S|$  is the arity of  $c_S$ .  $c_S \subset \prod_{x_j \in S} D_j$  specifies the *allowed* assignments for the variables of  $S$ .

## 1.2 Soft constraint networks

In this section, I introduce one generic soft constraint framework called valued constraint networks [Schiex et al., 1995]. I'm obviously biased in this choice by the fact that I have worked using this framework during the last decade. Writing about soft constraints is much easier for me using this framework. To compensate for this bias, I will try to relate this as clearly as possible to some existing alternate frameworks.

### 1.2.1 Valued constraint networks

In classical CN, each constraint specifies which tuple is authorized or not. Soft constraints aim at more flexibility: we want to be able, for each constraint, to specify to what extent one specific tuple violates (or satisfies) the constraint. It is therefore natural to transform constraints which are relations characterizing each tuple as either forbidden or authorized in classical CN into cost functions mapping each tuple to a specific level of violation. We therefore need a set  $E$  of such violation levels. This already give us a vague idea of what a soft constraint network could be:

DEFINITION 1 *A soft constraint network is a tuple  $\langle X, D, C \rangle$  where:*

- $X = \{x_1, \dots, x_n\}$  is a finite set of  $n$  variables.
- $D = \{D_1, \dots, D_n\}$  is the collection of the domains of the variables in  $X$  such that  $D_i$  is the domain of  $x_i$ .
- $C$  is a finite set of  $e$  soft constraints. A soft constraint  $f \in C$  is a function  $f_S$  on a set of variables  $S \subseteq X$ .  $S$  is the scope of the constraint.  $f_S$  maps tuples over  $S$  to elements of a specific set  $E$ , that is  $f_S : \prod_{x_i \in S} D_i \rightarrow E$ .

Now, obviously we want to try to violate constraints as little as possible. We shall therefore assume that the set  $E$  is ordered by the order  $\preceq$ . Also, considering that we want to extend classical networks, we must assume that the set contains two specific elements representing the complete violation (for forbidden tuples) and the absence of violation (for authorized ones). These two elements are denoted  $\top$  and  $\perp$  respectively.

Obviously a complete violation is stronger than any other level in  $E$  so  $\top$  should be maximum in  $E$ . Conversely, the absence of violation is the smallest possible violation and  $\perp$  should be minimum.

Now, if we are given a complete assignment  $t$  of  $X$ , then we will collect a set of levels: one level for each constraint  $f_S$  applied to  $t[S]$ . To minimize some “overall” level of violation, we must specify how such levels combine together. We assume that some binary, closed, commutative, associative operator  $\oplus$  can be used to combine all the levels into a global overall level. Associativity and commutativity are needed to capture the fact that the overall level just depends on the set of levels and not in a specific way used to combine them. The assumption of a closed operator is comfortable (and done without loss of generality, we may extend  $E$  if needed).

Since  $\top$  is associated to absolutely forbidden tuples, such that the use of a single such tuple is unacceptable,  $\top$  must be an annihilator for  $\oplus$  ( $a \oplus \top = \top$ ). Conversely, the absence of violation represented by  $\perp$  must not contribute in any way to the overall violation level and  $\perp$  should be an identity for  $\oplus$  ( $a \oplus \perp = a$ ).

A solution of our soft constraint network is a complete assignment  $t$  such that its overall level

$$level(t) = \bigoplus_{f_S \in C} f(t[S]) \neq \top$$

An optimal solution  $t$  is a solution such that  $level(t)$  is in some sense minimal: there is no other solution with a lower level. Because constraint networks are essentially used for design or decision problems where one wants to identify and use one solution, we further assume that the set  $E$  is totally ordered. This is a strong assumption, often realistic in practice.

Obviously, if some constraint  $f$  can be violated to level  $\alpha$  or  $\beta$  with  $\alpha \preceq \beta$  then, all other things being equal, the solution using the level  $\alpha$  cannot be worse than the solution using  $\beta$ : the operator  $\oplus$  must be monotonic ( $(a \preceq b) \rightarrow ((a \oplus c) \preceq_v (b \oplus c))$ ).

A totally ordered set  $E$  and equipped with such an operator is called a valuation structure [Schiex et al., 1995].

**DEFINITION 2** A valuation structure is a 5-tuple  $\langle E, \oplus, \preceq_v, \perp, \top \rangle$  such that:

- $E$  is a set, whose elements are called valuations, totally ordered by  $\preceq_v$ , with a maximum element  $\top \in E$  and a minimum element  $\perp \in E$ ;
- $E$  is closed under a binary operation  $\oplus$  that satisfies:
  - $\forall a, b \in E, (a \oplus b) = (b \oplus a)$ . (commutativity)
  - $\forall a, b, c \in E, (a \oplus (b \oplus c)) = ((a \oplus b) \oplus c)$ . (associativity)
  - $\forall a, b, c \in E, (a \preceq_v b) \rightarrow ((a \oplus c) \preceq_v (b \oplus c))$ . (monotonicity)
  - $\forall a \in E, (a \oplus \perp) = a$ . (neutral element)
  - $\forall a \in E, (a \oplus \top) = \top$ . (annihilator)

From an algebraic point of view, this structure can be described as a positive totally ordered commutative monoid, a structure also known as a positive *tomonoid* [Evans et al., 2001]. When  $E$  is restricted to  $[0, 1]$ , this is also known, in uncertain reasoning, as a triangular co-norm [Klement et al., 2000].

It should be noted that the annihilator property is not actually needed because it is a consequence of the other properties.

**EXERCICE 1** Show this.

A soft constraint network equipped with a valuation structure  $S$  is called a valued network:

**DEFINITION 3** A valued constraint network  $\langle X, D, C, S \rangle$ , where  $S$  is a valuation structure  $\langle E, \oplus, \preceq_v, \perp, \top \rangle$  is a soft constraint network using the valuation structure set  $E$  as the levels set.

Binary soft constraint networks can all be described using a variant of the so-called microstructural graph in classical CN (a multipartite graph):

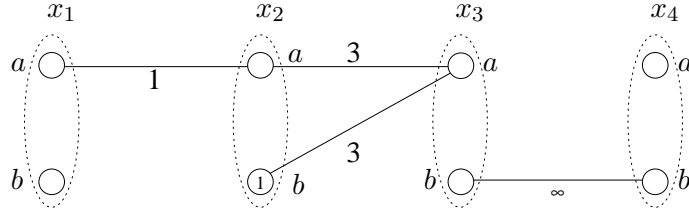


Figure 1.1: An example of soft CN with levels in  $\mathbb{N} \cup \infty$

- each value  $a \in D_i$  is represented by a vertex  $(i, a)$ .
- for  $a \in D_i, b \in D_j$  s.t.  $f_{ij} \in C$ , an edge connects the vertex  $(i, a)$  and  $(j, b)$  with weight  $f_{ij}(a, b)$
- unary constraints (if any) are represented as vertex labels.

When complete violation and complete satisfaction are defined, these graphs can be simplified as follows: edges with a level representing the absence of violation (identity) are omitted and levels which represent a complete violation (annihilator) are omitted (not the edge). This is incompatible with the usual classical CN representation (edge = allowed) but very comfortable.

We often assume when dealing with binary soft CN, that there is one unary cost function for every variable  $x_i$ . For simplicity it is denoted  $f_i$  instead of  $f_{\{x_i\}}$ . This is done without loss of generality since one can use a function mapping all values to  $\perp$  without changing the problem. Similarly, binary cost functions are denoted  $f_{ij}$ . We further assume the existence of a cost function with an empty scope (a constant)  $f_{\emptyset}$ . One should note that this gives an obvious lower bound on the level of an optimal solution of the network. If you don't like it, set it to  $f_{\emptyset}() = \perp$ .

## 1.2.2 Specific instances

By specifying the valuation structure used, many previously defined soft constraint frameworks can be defined:

- Classical CN: use  $E = \{t, f\}$  to denote respectively an authorized or unauthorized tuple.  $t = \perp \preceq f = \top$ .  $\oplus$  is  $\wedge$ .
- Conjunctive fuzzy CN [Rosenfeld et al., 1976]: use  $E = [0, 1]$  to denote membership degrees of a tuple to a fuzzy relation.  $\perp = 1 \preceq 0 = \top$ . The operator  $\oplus$  is min in the sense of the usual order on  $[0, 1]$  (i.e. max for the  $\preceq$  order). An optimal solution is a solution such that the minimum (usual order) membership degree used is maximum (usual order). Reversing the  $E$  scale gives the dual max-min optimization problem of Possibilistic CN [Schiex, 1992].

**EXERCICE 2** Consider you have an oracle that can solve classical CSP. Show that a conjunctive fuzzy constraint network  $P = \langle X, D, C \rangle$  can be solved using  $\log_2(n)$  calls to the oracle where  $n$  is the number of different levels used in the fuzzy network.

*Hint:* Consider the classical constraint network  $P^\alpha = \langle X, D, C^\alpha \rangle$  with the same variables and domains as  $P$  and such that for each fuzzy constraint  $f \in C$ ,  $C^\alpha$  contains a corresponding hard constraint  $c^\alpha$  whose relation contains only the tuples with satisfy  $c$  with a membership degree higher or equal to  $\alpha$  (using the usual order over  $[0, 1]$ )

This simple decomposition process can actually be used to extend most results on classical constraint networks to conjunctive fuzzy constraint network as long as these results rely on a property preserved by this slicing approach.

CSP	$E$	$\preceq$	$\top$	$\perp$	$\oplus$
<i>classical</i>	$\{t, f\}$	$t \preceq f$	$f$	$t$	$\wedge$
<i>additive</i>	$\mathbb{N}$	$\leq$	$+\infty$	$0$	$+$
<i>fuzzy</i>	$[0, 1]$	$\geq$	$0$	$1$	$\min$
<i>possibilistic</i>	$[0, 1]$	$\leq$	$1$	$0$	$\max$
<i>lexicographic</i>	$[0, 1]^*$	$\leq^*$	$\top$	$\emptyset$	$\cup$
<i>probabilistic</i>	$[0, 1]$	$\leq$	$1$	$0$	$1 - (1 - a)(1 - b)$

Table 1.1: A table of some specific instances

- weighted CN [Shapiro and Haralick, 1981, Larrosa, 2002]: if we assume that some fixed (not necessarily finite) integer cost  $k$  is considered as unacceptable, then we have  $E = \{0, \dots, k\}$ .  $\perp = 0 \preceq k = \top$ .  $\oplus = +$ . If  $k = 1$  we have a classical CN. If all constraints use only levels 0 and 1 then the problem is the Max-CSP problem [Freuder and Wallace, 1992].

EXERCICE 3 Consider the micro-structure in the Figure 1.1 seen as a weighted CN with  $k = 4$ . Compute the level of assignments  $(a, a, b, b)$ ,  $(a, a, a, a)$  and  $(b, a, b, a)$ .

- lexicographic CN [Fargier and Lang, 1993]: used to refine fuzzy CN. Elements of the set  $E$  are either multisets<sup>1</sup> of elements of  $[0, 1]$  or a specific element  $\top$ . The operation  $\oplus$  is multi-set union, extended to handle  $\top$  as an annihilator (the empty multi-set is the identity). The strict order  $\prec$  is the lexicographic order induced by the usual order  $>$  on multisets and extended to give  $\top$  its role of maximum element: let  $F$  and  $F'$ , 2 multisets and  $\alpha$  and  $\alpha'$  the smallest numbers in  $F$  and  $F'$ ,  $F \prec F'$  iff either  $\alpha > \alpha'$  or  $(\alpha = \alpha' \text{ and } E - \{\alpha\} \prec E' - \{\alpha\})$ . The recursion ends on  $\emptyset$ . The lexicographic ordering is total.

A fuzzy CN can be transformed in lexicographic CN by just replacing every membership degree in  $]0, 1]$  used by a multi-set containing just this element and replacing 0 by  $\top$ .

EXERCICE 4 Show that an optimal solution of the lexicographic CN is always optimal in the fuzzy CN and that the converse is false.

- probabilistic CN: used to model problem with constraints of uncertain existence.  $E = [0, 1]$ ,  $\top = 0 \succ 1 = \perp$ .  $a \oplus b = 1 - (1 - a) \times (1 - b)$ .
- Bayesian nets: use  $E = [0, 1]$ ,  $\top = 0 \succ 1 = \perp$ .  $\oplus = \times$ . Each conditional probability table in a Bayesian net is a cost function. Finding an optima solution is the MPE (maximum probability explanation) problem. Not all such problems are correct Bayesian nets.

EXERCICE 5 Show that any lexicographic CN can be transformed in an equivalent weighted CN (define equivalent). Similarly for a Bayesian net with rational probabilities.

EXERCICE 6 Consider the following dinner problem. I'm trying to decide what I will have for dinner with my friends. The main dish will be either fish or meat. The drink may be either water, Barollo or Greco di Tufo. Use you own preferences for the dish, the drink and the combination using first fuzzy CN and then weighted CN. What are the optimal solutions?

Note finally that multiple criteria are just multiple valuation structures that can easily be handled separately since they are independent. So, a multiple criteria soft CN is just a multiple VCSP.

<sup>1</sup>In a multiset, one element may be repeated.

### 1.2.3 Operations on cost functions

We can extend the usual operations and notions on classical constraints to cost functions. We consider three types of operations:

Consider a cost function  $f_S$  with scope  $S$ . Let  $x_i \in S$  and  $a \in D_i$ . The assignment  $a$  to  $x_i$  in  $f_S$  yields the soft constraint  $f_{S-x_i} = f_S[x_i = a]$  that maps any tuple  $t$  of  $\ell(S - x_i)$  to  $f_S(t \cdot (i, a))$ .

Consider two cost functions  $f_S$  and  $f_{S'}$ . The combination or join of the two cost functions with an operator  $\oplus$  is a soft constraint  $g_{S \cup S'} = f_S \oplus f_{S'}$  such that for any tuple  $t \in \ell(S \cup S')$ ,  $(f_S \oplus f_{S'})(t) = f_S(t[S]) \oplus f_{S'}(t[S'])$ . Obviously a solution of a soft CN  $(X, D, C, S)$  is therefore a tuple  $t$  such that  $(\bigoplus_{f \in C} f)(t) \neq \top$ .

Consider a cost function  $f_S$  and  $x_i \in S$ , projecting out or eliminating  $x_i$  from  $f_S$  yields a cost function  $f_{S-x_i} = f_S[S - x_i]$  (also noted  $f_S[-x_i]$ ) that maps any tuple  $t \in \ell(S - x_i)$  to  $\min_{a \in D_i} f(t \cdot (i, a))$ . More generally, we can project out or eliminate using any specific AC binary operator on levels and eliminate several variables at once.

A cost function  $f_S$  is said to be stronger than a cost function  $g_{S'}$  (denoted  $f_S \geq g_{S'}$ ) iff  $\forall t \in \ell(S \cup S')$ ,  $f_S(t[S]) \succcurlyeq g_{S'}(t[S'])$ . We also say that  $f_S$  is tighter than  $g_{S'}$  or that  $f_S$  implies  $g_{S'}$ . Indeed, in the classical case, if  $f_S$  is satisfied, then necessarily a weaker  $g_{S'}$  is satisfied too. If both  $f_S \geq g_{S'}$  and  $f_S \leq g_{S'}$  then the constraints are equivalent.

Using combination, these notions can be extended to arbitrary set of cost functions and constraint networks.

**EXERCICE 7** *Show that the combination of two cost functions is always stronger than each of the constraints. Show that projection out one variable from a cost function yields a weaker (or implied) soft constraint.*

Given a valuation structure  $\langle E, \oplus, \prec_v, \perp, \top \rangle$ , we will say that  $S$  is idempotent iff  $\oplus$  is idempotent ( $a \oplus a = a$ ). The only idempotent operator in valuation structures is  $\max$ .

**EXERCICE 8** *Show this.*

**EXERCICE 9** *Consider an idempotent soft constraint network  $P = (X, D, C, S)$  and  $f_S$ ,  $S \subset X$  an arbitrary cost function which is weaker than the constraint network  $P$ . Show that  $(X, D, C \cup \{f_S\})$  is equivalent to  $P$ .*

**EXERCICE 10** *Consider an arbitrary non-idempotent structure  $S$ , exhibit a network and a constraint implied by this network such that adding this constraint to the network yields a non equivalent network.*

When  $\oplus$  is strictly monotonic ( $\forall a, b, c \in E, (a \prec c), (b \neq \top)$  then  $(a \oplus b) \prec (c \oplus b)$ ), then  $S$  will be said strictly monotonic. If we consider two complete assignments  $t$  and  $t'$  such that for all  $f_S \in C$ ,  $f_S(t) \prec f_S(t') \neq \top$  and for some  $g_T \in C$ ,  $g_T(t) \prec g_T(t')$ , strict monotonicity guarantees that  $t$  will be preferred to  $t'$  i.e.  $level(t) \prec level(t')$ . This is therefore an attractive property from a rationality point of view. Note that strict monotonicity is incompatible with idempotency as soon as  $|E| > 2$  (see [Schiex et al., 1995]).

**EXERCICE 11** *Show this.*

### 1.2.4 Related frameworks

The design of a generic framework to capture specific instances of soft constraint frameworks was motivated by the desire to avoid repeated algorithmic work and also to understand why some problems (eg. weighted CN) are harder to solve than others (eg. classical or fuzzy). It is a matter of compromise between generality (maximize the number of framework covered) and specificity (stronger properties means more theorems, properties and algorithms).

Other frameworks have been defined with different compromises. The most famous one is the semiring network.

### 1.2.4.1 Semiring constraint networks

A semiring network is a soft constraint network that uses a c-semiring as the set of levels instead of a valuation structure. The main difference lies in the ability to deal with partially ordered set of levels (lattice based order). A second less important difference is that levels are considered as satisfaction levels (to maximize) rather than violation levels (to minimize).

DEFINITION 4  $S = \langle E, +_s, \times_s, \mathbf{0}, \mathbf{1} \rangle$  is a c-semiring when:

- A set  $E$  of satisfaction degrees.
- An operator  $+_s$  defines a partial order  $\preceq_s$  on the set  $E$ :  $a \preceq_s b$  iff  $a +_s b = b$  (ACI: associative, commutative, idempotent).
- a maximum element  $\mathbf{1}$  and a minimum element  $\mathbf{0}$ . Implies that  $\mathbf{1}$  is an annihilator for  $+_s$ ,  $\mathbf{0}$  a neutral element.
- an AC (associative, commutative) operator  $\times_s$  combines sat. degrees.  $\mathbf{0}$  is an annihilator for  $\times_s$ .
- $(a \times_s c) +_s (b \times_s c) = (a +_s b) \times_s c$  (distributivity).

This is more precisely an Abelian (or commutative) unitary semiring with the extra assumption of the existence maximum element and of the idempotency of  $+_s$  to capture lattice based ordering. One can show that c-semirings are strictly more general than valued networks because of this. Because of this additional ability, the central problem is to find undominated solutions (such that there is no better solution in the sense of  $+_s$ ).

However, there is the following general result:

EXERCICE 12 Show that any valued CN can be transformed in an equivalent totally ordered semiring CN and vice versa.

EXERCICE 13 Think of one c-semiring that is partially ordered. Show that it is a c-semiring. Build a small network using this c-semiring, list non dominated solutions. Compute the overall level of satisfaction of the network.

### 1.2.4.2 Constrained optimization

Most constraint solvers offer primitives for optimizing a specific integer variable. Since the central problem of soft constraint networks is to find an assignment that optimizes the specific criteria defined by the overall violation, it is natural to consider the expression of soft constraint networks as classical networks with a specific variable representing the optimized criteria.

Consider a valued constraint network  $\langle X, D, C, S \rangle$ ; beyond the original problem variables in  $X$ , we introduce one new variable  $x_S$  for each constraint  $f_S \in C$ . These extra variables have domain  $E$  (the set of possible valuation). Each constraint  $f_S \in C$  is transformed in a classical constraint  $c_{S'}$  whose scope  $S' = S \cup \{x_S\}$ . The set of authorized tuples of  $c_{S'}$  is obtained by taking every tuple  $t \in \ell(S)$  extended to  $S'$  by computing the associated semiring value  $f_S(t)$ . Finally, one extra variable  $x_c$  is introduced that represents the criteria. It is connected with all the  $x_S$  variables using one constraint specifying that  $\oplus x_S = x_c$ . It is easy to check that for any assignment  $t$  of  $X$ , the only possible value for  $x_c$  is the overall level of the assignment  $t$ . Maximizing it will lead to an optimal solution. This trick has been first proposed in a simplified form by [Petit et al., 2000]. It is used to model MAXSAT problems as pseudo-boolean problems in [de Givry et al., 2003].

This model has the advantage of offering a direct use of existing constraint propagation algorithms and makes the criteria optimized available as a variable which means that it is easy to impose constraints between criteria. But the 'reified' constraints defined, with increased arities will affect variable elimination algorithms (while it keeps the same good properties on valued CN as we will see). Actually, most efficient

algorithms for general valued networks (such as those that maintain some form of local consistency) propagate cost increases *per value*. It does not seem possible to represent such fine-grained information in the previous model. Ideally, some way of integrating both models elegantly has to be found.

Note that conversely, any constraint satisfaction problem with a criteria can obviously be represented as a soft constraint problem: hard constraints are kept and the criteria can be transformed in a combination of soft and hard constraint which involve the variables influencing the criteria (possibly using the above trick to get constraints of lower arities).

### 1.2.4.3 Others

Many other generic or specific frameworks have been defined. One can cite hierarchical constraint logic programming (HCLP) [Borning et al., 1989, Wilson and Borning, 1993], Partial CSP [Freuder, 1989, Freuder and Wallace, 1992] and general fuzzy constraint networks [Ruttkey, 1994] using arbitrary triangular norm for the combination operator. Although one can show that specific subclasses (or instances) of these frameworks are effectively covered by valued or semiring networks, we can only invite the reader to refer to the cited papers for more information.

EXERCICE 14 (FOR HOME) *After reading [Wilson and Borning, 1993], show that any HCSP using a global comparator with a specific form (whose generality will be maximized) can be transformed in an equivalent valued network.*

EXERCICE 15 (OPEN) *Then, establish a connection between locally or regionally better defined HCSP and semiring constraint networks (not advised unless you are strongly motivated, likely no strong result can be found here).*

Finally, as the satisfiability problem in propositional logic (SAT) is a subproblem of the constraint satisfaction problem, the problem MAXSAT [Papadimitriou, 1994] is clearly a subproblem of the weighted constraint satisfaction problem. This problem is known to be MAXSNP-complete and so without any polynomial time approximation scheme. This results directly applies to weighted CN and general valued CN optimisation problem.

## 1.3 Search algorithms

Many queries can be associated with a soft CN:

- compute the cost of an *optimal* (non dominated) solution;
- *find one/all optimal* (non dominated) solutions;
- find a *sufficiently good solution* (cost less than  $k$ );
- prove that a given value/tuple is *not used* in any (optimal) solution;
- transform a soft CN into an *equivalent* but simpler soft CN. . .

Here we consider algorithms that essentially look for one optimal solution or hopefully good solution. An algorithm that guarantedly produces an optimal solution is said complete. We start by a class of complete algorithms.

### 1.3.1 Branch and bound

A first simple approach to solving soft CN is to used a branch and bound algorithm instead of a backtrack algorithm. The algorithm explores a tree of soft CN. The root is our initial problem  $P = (X, D, C, S)$ . Imagine we already know a solution of level  $\alpha$ , then we are only interested in solutions with a strictly lower level:  $\alpha$  is an upper bound on the level of an optimal solution. We can truncate the set  $E$  by replacing all

valuations larger than  $\alpha$  by  $\alpha$  in the network, in  $E$  and in the map of  $\oplus$ . If we set  $\top = \alpha$  in the valuation structure, we obtain a valuation structure.

The sons of a given node is obtained by choosing one variable  $x_i$  of the problem and assigning it one value of its domain. In the obtained subproblem, all the constraints involving  $x_i$  have their scope reduced using the assignment operator. Eventually, some scopes will become empty and contribute to the value of  $f_\emptyset$ . Each node  $n$  in the tree is associated with a tuple defined by the sequence of assignments done (on so-called past variables) and by the associated reduced problem with a specific  $f_\emptyset$ .

The branch and bound algorithm avoids a complete tree exploration by exploiting a local lower bound  $lb(n)$  on the level of an optimal solution of the root problem that can be found under the current node  $n$  by extending the current tuple  $t$  to a complete tuple. This is a crucial component of branch and bound: one aims at an easily computed but strong lower bound (as large as possible). In our case, the current value of  $f_\emptyset$  offers a trivial lower bound. Stronger lower bounds will be considered later.

At a given node  $n$ , if  $lb(n)$  is larger than or equal to the upper bound  $\top$  then there is no point in exploring the subtree: all complete assignments will have a value larger or equal to the upper bound. Conversely, in a depth first search of the tree, if a complete assignment is reached, it is a new solution with an overall level equal to the current  $f_\emptyset$  and strictly lower than the know upper bound  $\alpha$ : we can again truncate the valuation structure by setting  $\top$  to this new value, using the level of the new solution as a new stronger upper bound.

Such a branch and bound algorithm for arbitrary valued networks was initially presented in [Schiex et al., 1995] and is closely related to the branch and bound procedure for MaxCSP introduced by [Freuder and Wallace, 1992]. The two objects  $\top$  and  $f_\emptyset$  provide an elegant way of representing the usual upper and lower bounds of a minimizing branch and bound. This nice representation has been proposed by [Larrosa, 2002]. Other branching or exploration strategies are obviously possible. The procedure depends also on variable and value ordering heuristics, as in classical CN. The same general principles apply. See [de Givry et al., 2003] for more information on heuristics.

---

**Algorithm 1:** Branch and bound

---

```

Function  $DFBB(X, D, C)$ 
  if  $X = \emptyset$  then  $\top = c_\emptyset$ ;
  else
     $x_j := \text{selectVar}(X)$ ;
    foreach  $a \in d_i$  do
       $\forall f \in C$  s.t.  $x_j \in S, f \leftarrow f[x_j = a]$ ;
      if  $f_\emptyset \preceq \top$  then  $DFBB(X - x_i, D - D_i, C)$ 

```

---

EXERCICE 16 *Apply this algorithm on the MaxCSP defined by the inconsistent three queens problem. Describe the current problem at each node.*

The lower bound  $f_\emptyset$  used in this algorithm can be replaced by any other available lower bound. The history of soft constraint algorithms is extremely rich in proposals for this bound. A first simple improvement can be obtained by taking into account the fact that before being reduced to empty scope constraints, assigned constraints will first reduce to unary constraints. Since every variable in the reduced problem must be assigned, for each variable  $x_i \in X$  we will have to account at least for an extra  $\min_{a \in D_i} f_i(a)$ . So a better lower bound is defined by  $lb_{fc}(X, D, C) = f_\emptyset \oplus \bigoplus_{x_i \in X} \min_{a \in D_i} f_i(a)$ . This is related to an extension of forward checking called Partial Forward checking (PFC) [Freuder and Wallace, 1992] extended to arbitrary valued CN in [Schiex et al., 1995].

EXERCICE 17 *Apply this algorithm on the MaxCSP defined by the inconsistent three queens problem. Describe the current problem at each node.*

Several stronger lower bounds, some of them being NP-hard to compute, have been proposed since in [Schiex et al., 1995, Wallace, 1995, Larrosa and Meseguer, 1999, Larrosa, 2002, Meseguer et al., 2001,

Affane and Bennaceur, 1998, Régim et al., 2001], most often for weighted CN. All these lower bounds try to take into account the constraints of arity above one in the current subproblem. Many of these have never been tested in practice and other are clearly subsumed by lower bounds directly induced by local consistency enforcing (which will be later introduced). Most of these lower bounds are defined on weighted CN. On idempotent networks, stronger lower bounds are easily produced (see [Rosenfeld et al., 1976, Snow and Freuder, 1990, Schiex, 1992]).

### 1.3.2 Russian Doll Search

One class on NP-hard yet effective lower bounds has been introduced in the Russian Doll Search algorithm [Verfaillie et al., 1996]. We assume that we have a binary valued CN. At any point in the tree, the subproblem to solve can be decomposed in two parts: one part is a set  $C_1$  of unary constraint plus the empty scope constraint produced by constraint assignments, the rest  $C_2$  is composed of binary constraints with no involving only variables in  $X$ , the set of currently unassigned variables. A lower bound on the first set of constraints is easily obtained by the  $lb_{fc}$  lower bound. But all other constraints, which are original constraints of the problem, are ignored... A simple way to improve our bound is to effectively solve the CN defined by this second set of constraints. If  $level(X, D, C_2)$  denotes the level of an optimal solution of the problem  $(X, D, C_2)$ , then obviously  $lb_{rds}(X, D, C) = level(X, D, C_2) \oplus lb_{fc}(X, D, C_1)$  is probably a much stronger lower bound. In order to compute  $level(X, D, C_2)$ , we will inductively use a branch and bound using the  $lb_{rds}$  lower bound!

But how solving such a subproblem at each node may speed up the search? The essential idea is to use a fixed variable ordering  $x_1, \dots, x_n$  for variable assignment (using always the same variable at the same depth). In this case, at fixed depth  $l$ , the subproblem  $P_l = (X, D, C_2)$  to solve will be always the same and defined only by the set of unassigned variables  $\{x_l, \dots, x_n\}$  and all the relevant binary constraints. It therefore suffices to solve only  $n$  such subproblems: we first solve the problem defined just by the set of variables  $\{x_n\}$ . This one is trivial and allows to compute  $level(P_n)$ . We then solve increasingly large problems involving one extra variable at each step. Because all the subproblems needed to compute  $lb_{rds}$  at a given iteration have already been solved on previous iterations, we can directly use them in the current branch and bound iteration. We have to solve  $n$  problems of increasing size and the last one is our original complete problem. Despite the increased number of searches, the stronger lower bound may strongly speed up the search by several orders of magnitude. See [Verfaillie et al., 1996] with experiments on satellite scheduling problems. Several extra tricks can be used to speed up the search by exploiting the information returned by previous iterations (solutions...). This algorithm has been later sophisticated by [Meseguer and Sanchez, 2001, Pedro Meseguer et al., 2002]. In the specific case where all constraints but unary constraints are hard, the algorithm has a nice and efficient constraint programming formulation known as lightRDS [T. Benoist and M. Lemaître, 2003].

EXERCICE 18 *Show that despite the repeated number of searches, a ratio of the number of the nodes explored in the worst case situation for  $n$  searches vs. the worst case for a single one is not very large.*

### 1.3.3 Local search

Local search techniques are widely applicable combinatorial optimization methods. The general idea of local search is to start from a potential solution  $t$ , and to try to *locally modify*  $t$  into  $t'$ , close to  $t$  but potentially better and repeat until satisfied.

Compared to the previous methods, their main characteristics is that they are not complete (cannot offer a guarantee to reach an optimum). Applied to a pure satisfaction problem, these methods still have the nice property that when a solution is found, it is known to be “optimal” but for soft constraint, the optimization task requires to both find an optimum (in the FNP-complete class of problems) and to prove that no better solution exists (a co-FNP problem for which no short certificate exists under the usual assumption that P is not NP). Nevertheless, these methods have often good results and are reasonably easy to implement which makes them attractive when complete methods are unable to tackle difficult problems (there are other alternatives in this case, among which relaxed version of complete search can also be considered, see for example [de Givry and Jeannin, 2004]).

We don't intend to give here a course on local search. There are so many different approaches that a simple course on the topic would require a lot of time. Just to cite a few, consider methods like simulated annealing, Tabu search, variable and large neighborhood search, greedy random adaptive search, genetic algorithms, ant colony and bee swarm optimization, Go with the winners... There are both very good books [Aarts and Lenstra, 1997] and very good background papers [Christian Blum and Andrea Roli, 2003] on this topic.

So we just highlight the essential principles and indicate how it can be used to solve soft constraint problems and simply combined with complete search methods.

In constraint networks, local search methods explore the space of complete assignments (potential solutions) by incrementally modifying one or several potential solutions (which may sometimes be set of complete tuples represented concisely for example by a partial tuple [Cédric Pralet and Gérard Verfaille, 2004]). We restrict ourselves to the simple case where the algorithm maintains a current complete tuple  $t$  of the soft CN  $(X, D, C, S)$ . A *move* is an elementary operation that returns a *neighbour* of  $t'$  and the set of all neighbors of  $t$  is its neighborhood. A trial is a succession of moves and a local search a succession of trials. The criteria to optimize (minimize here) is its level and the general idea is to favor moves that will decrease the level of the tuple, yet avoiding to get stuck in local minima.

A traditional move in constraint networks is to pick a variable and to change the assignment of the variable in the tuple  $t$  (or of several variables if large neighborhoods are considered). A general schema for local search capturing a reasonable percentage of the existing methods is given in Algorithm 2. The schema is parametrized by the number of allowed moves and trial (*Max-trials*, *Max-Moves*) and by three procedures that will generate the first putative solution (**NewSolution**), generate a possible move from  $t$  (**ChooseNeighbor** ( $t$ )) and decide if the move is executed based on the change of the criteria it induces (**Accept?** ( $\delta$ )). A basic method, greedy search, is defined by:

- **ChooseNeighbor** ( $x$ ): choose randomly a best neighbor (greedy).
- **Accept?** ( $\delta$ ) : *true* (We always accept)

which get rapidly stuck in local minima but is greatly improved using random moves with a fixed small probability.

---

**Algorithm 2:** A local search schema

---

```

LocalSearch ();
 $x^* \leftarrow$  NewSolution ();
for  $t = 1$  to Max-Trials do
   $x \leftarrow$  NewSolution ();
  for  $m = 1$  to Max-Moves do
     $x' \leftarrow$  ChooseNeighbor ( $x$ );
     $\delta \leftarrow (\varphi(x') - \varphi(x))$ ;
    if  $\varphi(x') < \varphi(x^*)$  then
       $x^* \leftarrow x'$ ;
    if Accept? ( $\delta$ ) then
       $x \leftarrow x'$ ;
return Nothing better than ( $x^*, \varphi(x^*)$ )

```

---

Local search methods being brute force methods, special care must be taken to make the space exploration as efficient as possible:

- a solution should be simple to represent (here a tuple)
- the application of a move should be typically *constant time*
- the change in the criteria after a move should be *incrementally computed* from the previous one (constant time).

Specific languages have been designed to facilitate this [Laurent Michel and Van Hentenryck, 2000]. There is a very large bibliography on application of local search to various combinatorial problems some of which being direct instances of soft (often weighted) constraint networks. We just give here some references directly related with soft constraints [Lau and Tsang, 2001, Galinier and Hao, 1997, Samir Loudni and Patrice Boizumault, 2003].

There are many different way to integrate local search with complete branch and bound based search. The most simple (and yet effective) method is to use local search techniques to provide the initial upper bound for optimization. But many other combination exists: complete methods can be used to explore large neighborhoods more efficiently [Samir Loudni and Patrice Boizumault, 2003], can be used as guide during tree search...

### 1.3.4 Experimenting with soft CN

As in the classical CN case, benchmarking is one possible way to validate the efficiency of a new solving algorithm. Benchmarking can be done on several types of problems: (often simplified) real problems, academic problems and random problems. In this section, we just consider random networks, but see §1.6.1 for resources on other types of problems.

Existing random models for soft CN I'm aware of are in fact classical SAT or classical CN generators. Because the generators are naturally capable of generating unsatisfiable instances, it suffices to generate such problems and solve the MaxCSP/MaxSAT problem where the aim is to find an assignment that minimizes the number of violated constraints/clauses. It is therefore usual to generate problems far from the phase transition region induced by these generators.

The usual observation is that there is no phase transition in MaxSAT/MaxCSP random problems. Indeed, as far as the problem is inconsistent, a complete solver has to solve both an NP problem (finding an optimal solution) and a co-NP problem (proving optimality) and will therefore always work around some phase transition for a pure decision problem (is there an assignment that violates less than  $k$  constraints/clauses). Problems are therefore increasingly difficult as the size of the problem grows.

However, some side issues may blur this figure:

- bounded arity SAT generators: since 3SAT is NP-complete, a lot of effort has been spent on 3SAT problems. In soft CN, even Max-2SAT is NP-complete so we can even consider 2SAT problems. Usually, for a fixed number of variables, the number of clauses is increasingly augmented. But many generators avoid generating duplicate clauses. With  $n$  variables, this means that the number of clauses is bounded and that ultimately when we increase the number of clauses, the problem will be trivial and formed of all possible  $k$ -clauses on  $n$  variables. These problems are usually trivially solved as far as a non stupid bounds are used. Therefore, the complexity seems to lower on very tight problems but this depends on the algorithm.

It is therefore a good idea to use generators that allows for duplicate clauses. Note also that duplicate (weighted) clauses MaxSAT is  $FP^{NP}$ -complete whereas without repeated clauses, it is only in  $FP^{NP[\log n]}$  [Papadimitriou, 1994].

- random binary CN: here, for a fixed number of variables  $n$  and a fixed number of values  $d$ , one may explore either the constraint tightness or the constraint density parameters. If the constraint tightness is used, then ultimately, all constraints will be hard constraints forbidding everything and any non trivial lower bound will detect this immediately. This is different with constraint density (a complete problem will be very hard usually).

So one should not restrict experimental studies to only constraint tightness.

This has lead to the observation of unusual phase transitions in MaxCSP Larrosa and Meseguer [1996].

## 1.4 Inference algorithms

In propositional logic or classical CN, inference algorithms are in charge of producing formulas which are logical consequence (if the inference system is sound) of a set of formulas. It is said complete if it

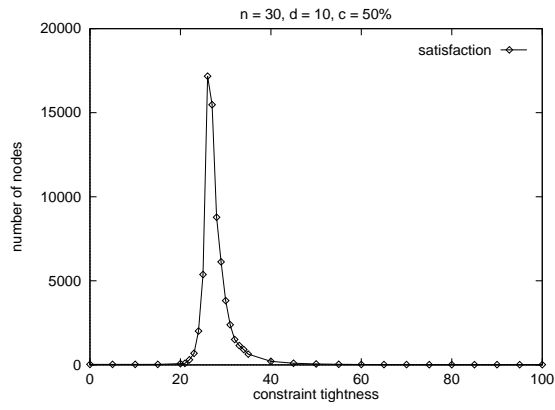


Figure 1.2: Classical CSP phase transition

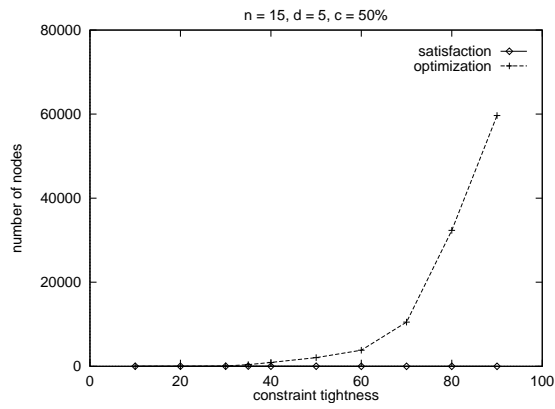


Figure 1.3: Finding an optimal solution

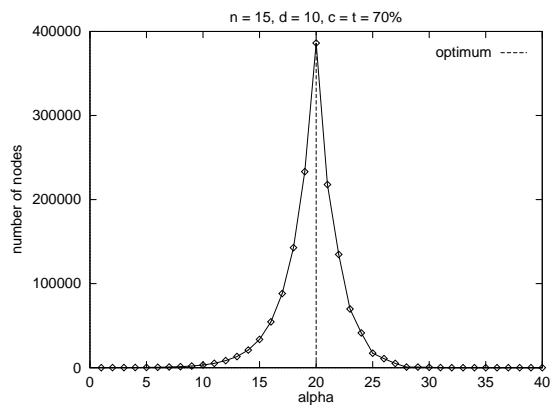


Figure 1.4: Is there an assignment that violates less than  $k$  constraints

can produce arbitrary logical consequences. In a weaker form, one could ask for just the detection of inconsistent set of clauses/constraints by production of the empty clause/constraint.

Using cost functions, what we may aim at is the production of cost functions which are “implied” by a soft constraint network (see §1.2.3. Among these, the strongest zero-arity constraint implied by a soft CN is specifically interesting since it gives the level of the network. We consider that an inference mechanism is complete for valued CN when it is capable of producing this constraint. This is a very weak requirement and stronger definitions can certainly be considered.

EXERCICE 19 *What alternative definition of complete could you propose ? Think that it is not possible to add an implied constraint to a non-idempotent CN w/o changing its semantics (see the incomplete inference section too).*

## 1.4.1 Complete inference

In this section we consider inference systems which are based on *variable elimination*. This very general technique has been extensively used in several domains (for linear equations, relational databases and cost function processing among others) and it is a complete inference mechanism according to our definition. This is a quite old result actually for weighted CN since it already appears in [Bertelé and Brioshi, 1972] and forms the class of *non serial dynamic programming* algorithms.

Although these results clearly generalize the corresponding results in classical CN, the extension to cost function is straightforward and we therefore just present the essential notions and will see some examples. To give a different light on these results, we use the historical denomination used in the seminal book of [Bertelé and Brioshi, 1972].

### 1.4.1.1 Variable elimination

Consider an arbitrary valued CN (actually, this approach works for a class of problems that properly includes c-semiring structures and includes counting problems, see [Shenoy, 1991]).

Consider variable  $x \in X$ , we note  $K_x = \{f_S \in C, x \in S\}$  and  $L = (\cup_{f_S \in K_x} S) - \{x\}$ .  $K_x$  is the set of all cost functions involving  $x$  and  $L$  is just the set of all the variables involved in one of these cost functions with  $x$  removed (eliminated).

Consider the cost function defined by  $\oplus_{f_S \in K_x} f_S$  of all constraints in  $K_x$  and project out  $x$  from it getting  $f^x = (\oplus_{f_S \in K_x} f_S)[-x]$ .

We know this constraint is implied by  $(\oplus_{f_S \in K_x} f_S)$  and more that for any assignment  $t$  of the variable in  $L$ , there is an assignment  $t'$  extending  $t_L$  on  $x$  such that:

$$(\oplus_{f_S \in K_x} f_S)[-x](t) = (\oplus_{f_S \in K_x} f_S)(t')$$

Let's call all such tuples  $t'$  the supports of the tuple  $t$ .

Now, if we remove all the cost functions in  $K_x$  from the network, remove variable  $x$  from  $X$  and instead put back the new cost function (of scope  $L$ )  $f^x = (\oplus_{f_S \in K_x} f_S)[-x]$ , we get a network with the same optimal cost.

We can do this repeatedly. This is variable elimination. At the end we get a zero-arity constraint whose level is the level of the network (by induction).

EXERCICE 20 *Apply this to the inconsistent three queens problem. Pay attention to keep all the intermediate cost functions. Now, trace back all the supports inductively from the final zero arity constraint. You have all the optimal solutions.*

Defined for arbitrary graphical models, this algorithm is also called bucket elimination [Dechter, 1997].

- fix a *variable ordering*  $x_1, \dots, x_n$
- one *bucket* per variable, from last to first: contains all constraints involving the variable (not already in a bucket).

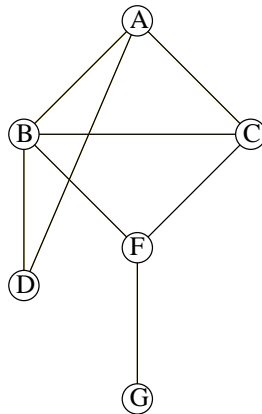


Figure 1.5: A simple graph.

- process from last to first:
  1. join all constraints  $K$  in the bucket
  2. eliminate the current variable by projecting it out.
  3. put the projection in the first bucket that contains one variable of  $L$ .

What is the complexity of this process?

- *Time complexity*: dominated by the time to compute the largest  $\bowtie K$ . Exponential in  $|L| + 1$  for the largest  $L$ .
- *Space complexity*: dominated by the space to store the largest projection  $(\bowtie K)[L]$ . Exponential in  $|L|$  for the same  $L$ .

Can we influence this maximum  $|L|$  by the order used to eliminate variables? Yes, and this depends intimately on the structure of the graph of the problem (binary networks). These results have also been extended to hypergraphs.

EXERCICE 21 Consider the graph in Figure 1.5. What is the maximum size of the set  $L$  over the complete elimination for orders  $A, C, B, F, D, G$  and  $A, F, D, C, B, G$ ?

This leads to the definition of width, induced graph and induced width.

For a graph  $G = (V, E)$ , an order of vertices  $d$ :

- width of a vertex: number of connected predecessors (parents)
- width of ordered graph: maximum width of a vertex
- width of graph: min. width over all ordering

EXERCICE 22 Compute this on the previous graph using order  $A, C, B, F, D, G$  and  $A, F, D, C, B, G$ .

The induced graph simulate the elimination algorithm: when we eliminate we induce a new constraint that is added to the problem. We represent this constraint using a clique added to the graph. Therefore, by processing vertices from the last to the first, we connect all the parents of a vertex together (this is the current set  $L$ ).

The width on an induced graph is equal to the set of the largest  $L$  we will deal during elimination (also known as the  $k$ -tree number, max-clique size-1, tree width...). the bad news is that finding a min-induced width ordering is NP-hard.

EXERCICE 23 Use this elimination algorithm on a linear network (a network forming a line). What complexity do you get starting from an extremity? Now on a tree ?

To better understand how a graph with treewidth  $k$  looks, there is a simple characterization as partial  $k$ -trees. This does not help finding the right  $k$ .

A  $k$  tree is inductively defined as a  $k$ -clique, or by the addition of a new vertex to a  $k$ -tree, connecting it to all vertices of a  $k$ -clique in it.

EXERCICE 24 Draw a 2-tree, a 3-tree.

**Notes** This class of algorithms has been repeatedly rediscovered, improved, refined. It is especially applicable to all graphical models which are frameworks with variables, domains and local functions. CN, soft CN, Bayesian nets... are examples of graphical models. These algorithms can solve decision, optimization and counting problems. Some references in this area are [Bertelé and Brioshi, 1972, C. Beeri et al., 1983, S.L. Lauritzen and D.J. Spiegelhalter, 1988, Dechter and Pearl, 1989, Shenoy, 1991, Dechter, 1999, Bistarelli et al., 1997]. A very nice example of a purely academic use of variable elimination appears in [J. Larrosa and E. Morancho, 2003].

Instead of eliminating just one variable after the other, it is also possible to eliminate several variable in a row. This is called block by block elimination in [Bertelé and Brioshi, 1972] and is related to cluster tree elimination algorithms [Dechter, 1999]. This can improve the spatial complexity.

A related class of work combines the exploitation of the graph structure (a tree-decomposition of the graph) similarly to the variable elimination algorithm, the use of tree-search to solve each combined constraint in this decomposition together with local strong bounds as above. The first reference in the area is probably [Freuder and Quinn, 1985] pseudo-tree search algorithm which is defined only on classical CN (but is easily extended to values CN) and was actually used in conjunction with Russian Dolls Search in [Javier Larrosa et al., 2002]. It was first considered in its general form for discrete integration tasks in Bayesian nets by [Adnan Darwiche, 2001] and then considered with additional propagation as the Back-track Tree Decomposition for optimization by [Cyril Terrioux and Philippe Jégou, 2003]. It has recently received additional interest for optimization by [Radu Marinescu and Rina Dechter, 2005].

## 1.4.2 Incomplete inference

When the graph structure is not ideal for complete inference (too space or time consuming), we may try to limit ourselves to some less ambitious (having only a lower bound on the optimum) but more tractable problem. We call this incomplete inference.

### 1.4.2.1 Mini-buckets

Since the computation of the combination of all constraints in  $K_x$  is very expensive, one can instead of computing

$$f^x = (\oplus_{f_S \in K_x} f_S)[-x]$$

partition the set  $K_x$  in several subsets  $K_x^i$  and just compute a family of cost functions

$$f_i^x = (\oplus_{f_S \in K_x^i} f_S)[-x]$$

If the number of variables involved in each “mini-bucket” is small, the combination and the projection are efficient and generate only small arity cost functions. Then, we can perform variable elimination using the family  $f_i^x$  instead of  $f^x$  (after removing  $x$  and the constraints in  $K_x$  we just put the  $f_i^x$  back). Since:

$$\bigoplus f_i^x \leq f^x$$

because taking the minimum at the latest is always better, the problem obtained has a level which is less than (or equal to) the level of the original problem. Doing this repeatedly will inductively produce a lower bound available as the final  $f_\emptyset$  [Dechter, 1997]. If  $k$  is the maximum number of variables in every bucket then this algorithm is in time  $O(e.d^k)$  and space  $O(e.d^{k-1})$ . For  $k = 2$  on binary constraints, this is related to directional arc consistency enforcing (see next section).

### 1.4.2.2 Soft local consistency

In classical CSP, local consistency properties are essential components of the toolbox for solving CN. A local consistency property is defined as a relaxation of consistency which can be checked in polynomial time. It is accompanied by corresponding “filtering” or “enforcing” algorithms that compute, in polynomial time, and from any given CN, an equivalent network that satisfies the property. Probably the most famous local consistency level is the so-called arc consistency level. The essential use of such filtering algorithms lies in the fact that if this equivalent problem (called the locally consistent closure) is empty, then the initial problem is obviously inconsistent too.

The same motivation exists for extending local consistency to soft constraints: the hope that the equivalent locally consistent problem may provide a better lower bound on the level of consistency of the network than the initial value of  $f_{\emptyset}$  (or  $lb_{fc}$ ).

In the sequel we consider binary soft constraint networks  $\langle X, D, C, S \rangle$ . This restriction is done only for the sake of simplicity since most results have been originally presented for arbitrary arities.

**A first operational approach** A first approach to extend local consistency is operational: by directly extending the fundamental operations that underlie local consistency to soft constraints, one may hope to obtain properties and algorithms that will apply to soft networks.

Consider a classical network  $(X, D, C)$ , variable  $x_i \in X$  is said to be arc-consistent relative to a constraint  $R_S$  s.t. that  $x_i \in S$  iff for every value  $a$  in  $D_i$  there exists a tuple  $t \in R_S$  such that  $t[x_i] = a$  and for every  $x_j \in S, t[x_j] \in D_j$ .  $t$  is said to be a support of  $a_i$  on  $R_S$ . A variable  $x_i$  is arc-consistent when it is arc-consistent w.r.t. all constraints whose scope is a proper superset of  $\{x_i\}$ . The network  $(X, D, C)$  itself is arc-consistent when all its variables are arc-consistent.

In other words, a variable  $x_i$  is arc consistent (AC) w.r.t.  $R_S$  when  $R_i \subset (R_S \bowtie_{x_j \in S, j \neq i} R_j)[x_i]$ : all the information provided by one non unary constraint  $R_S$  and the domains of the other variables involved in  $S$  does not provide any new information on  $x_i$  domain. This definition only uses constraint composition (*Join* that corresponds to combination  $\oplus$ ), projection, constraint ordering by inclusion which we all already extended to soft constraints.

**DEFINITION 5** Given a soft constraint network  $P = \langle X, D, C, S \rangle$ , a variable  $x_i \in X$  is arc-consistent w.r.t. a constraint  $f_S$  iff for every value  $a_i \in D_i$ ,  $f_i \geq (f_S \oplus_{x_j \in S, j \neq i} f_j)[x_i]$ .  $P$  is arc-consistent when all its variables are arc-consistent w.r.t. to all the constraints in  $C^+$  that involve it.

It is natural to consider an associated arc-consistency enforcing algorithm that should transform an original soft network  $P$  in an equivalent network (in the sense of  $\geq$ , see § 1.2.3) which is also arc-consistent. This algorithm works by considering all variables  $x_i$  that violate the arc-consistency condition ( $f_i < (f_S \oplus_{x_j \in S} f_j)[x_i]$ ). In this case, one simply enforces  $f_i \leftarrow f_i \oplus (f_S \oplus_{x_j \in S, j \neq i} f_j)[x_i]$  as the **Revise** procedure does in the classical case. Note that this can only increase the levels of values in  $f_i$  (and thus our lower bound  $lb_{fc}$ ). This is done iteratively until quiescence (if any). A naive version of the algorithm is presented in Algorithm 3. The boolean  $Q$  is used to represent quiescence.

---

**Algorithm 3:** Enforcing arc consistency in soft idempotent constraint network

---

```

 $Q \leftarrow false;$ 
while  $\neg Q$  do
   $Q \leftarrow true;$ 
  foreach  $x_i \in X$  do
    foreach  $f_S \in C$  s.t.  $x_i \in S, |S| > 1$  do
       $f \leftarrow f_i \oplus (f_S \oplus_{x_j \in S, j \neq i} f_j)[x_i];$ 
      if  $f \neq f_i$  then
         $f_i \leftarrow f;$ 
         $Q \leftarrow false;$ 

```

---

This definition and the associated enforcing procedure have been initially formulated in a more general settings, for arbitrary local consistencies (not only arc or  $k$ ) in [Bistarelli et al., 1995, 1997] with the following positive result:

**THEOREM 1** *If  $\times_s$  is idempotent then the algorithm terminates and yields a unique equivalent arc-consistent soft network.*

By uniqueness we mean that the network obtained at the end is unique and independent of the order of processing of variables/constraints in Algorithm 3. The condition of idempotency is sufficient but can be slightly relaxed for termination. However, it is a necessary condition to guarantee equivalence.

**EXERCICE 25** *Show that the constraint  $(f_S \oplus_{x_j \in S, j \neq i} f_j)[x_i]$  is implied by the network. Consider a non-idempotent structure and show that just one iteration can strictly increase the level of some complete assignments.*

Note that these results apply without any assumption of total order and therefore, beyond fuzzy constraint networks (which is the only totally ordered idempotent c-semiring), they apply to all partially ordered c-semiring structures such as multi-criteria optimization with only idempotent criteria or, more significantly, to set lattices based criteria. However, a lot of real problems do not rely on idempotent operators which suffer from insufficient discrimination and rather rely on frameworks such as weighted or lexicographic constraint networks.

### 1.4.2.3 Dealing with more operators

It is easy to see that local consistency defined as above cannot work in non idempotent networks because adding some implied constraint may increase the level of some complete assignments (equivalence is lost).

How can we avoid this increase and preserve equivalence? To achieve this, we need to be able to compensate for the increase of level at the unary level by a corresponding decrease at the binary level. We therefore require a new operation to be able to “subtract” some level from an higher level. This ability was first considered by [Schiex, 2000]:

**DEFINITION 6**

Several examples of fair and unfair structures are given in [Cooper and Schiex, 2004] but all the usual instances of totally ordered c-semiring are fair or can be plunged in a fair equivalent structure. For example, in fuzzy networks using  $\times_s = \min$ , the difference is also min since if  $\alpha \succ_s \beta$ , then  $\min(\alpha, \min(\alpha, \beta)) = \beta$ . In weighted networks where  $\times_s$  is the bounded addition defined by  $\alpha +^k \beta = \min(k, a + b)$ , the difference is  $-^k$  defined by:

$$\alpha -^k \beta = \begin{cases} \alpha - \beta & : \alpha \neq k \\ k & : \alpha = k \end{cases}$$

This allows us to define a new fundamental operation on cost functions called extraction:

Let  $f_S$  and  $f_{S'}$  be two cost functions such that  $f_{S'} < f_S$ , the extraction of  $f_{S'}$  from  $f_S$  is the cost function  $g_{S \cup S'} = f_S \ominus f_{S'}$  such that for any tuple  $t \in \ell(S \cup S')$ ,  $(f_S \ominus f_{S'})(t) = f_S(t[S]) \ominus f_{S'}(t[S'])$ .

We have that  $f_{S'} \oplus (f_S \ominus f_{S'})$  is equivalent to  $f_S$ . It is now possible to add an implied constraint to a network and then extract it from its source to preserve equivalence. We have all the elements to define local consistencies on fair structures. This has been done for arbitrary arity constraints in [Schiex, 2000, Cooper and Schiex, 2004] and for  $k$  consistency in general in [Martin Cooper, 2005]<sup>2</sup>. For simplicity, because it is a frequently used case in practical problem, which is very general, because the bibliography contains many results defined on this case and whose extension to the general case requires more thinking, we restrict ourselves in the rest of the section to weighted CN<sup>3</sup>

<sup>2</sup>Note that these papers use two operations called “Project” (which combine projection and extraction) and “Extend” (which combines combination and extraction) instead of just using extraction and are relatively involved because they aim at maximum generality (and are effectively able to tackle arbitrary fair valued CN).

<sup>3</sup>Weighted CN have no idempotent elements (such that  $a \oplus a = a$ ) beyond  $k = \top$  and  $0 = \perp$ . This simplifies a lot the general AC enforcing algorithms and the definition of associated properties which include additional properties to ensure that idempotent elements are propagated enough to ensure equivalence with AC on classical and fuzzy cases.

Our first (doomed to fail) tentative is to reuse the previous definition of AC and just be smarter when we enforce it by compensating for the combination of any new implied constraint by extracting it first from its source (which must be made explicit). Consider the network in Figure 1.6(a) using an infinite upper bound. It has two variables  $x$  and  $y$  with two values  $a$  and  $b$ . There is one binary constraint  $f_{xy}$  and one non trivial unary constraint.

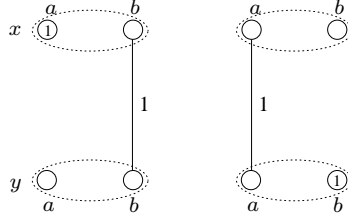


Figure 1.6: Two simple equivalent weighted CN with  $k = \infty$

Following our previous operational method we can combine  $f_{xy}$  and  $f_x$  and project on  $y$ . We get a non trivial cost function that maps  $b$  to 1. So we first replace  $f_{xy}$  by  $f_{xy} \oplus f_x$  and clear  $f_x$  (making the source of the projection explicit), then replace  $f_y$  by  $f_y \oplus (f_{xy} \oplus f_x)[y]$  (combine the implied constraint) then to compensate we extract it from its source. We get a network which is symmetrical w.r.t. to the initial network. Obviously, we are therefore entitled to apply the same process symmetrically again and again: our algorithm does not terminate. This has been first considered in [Schiex, 2000] and later called Full Arc Consistency [Simon de Givry et al., 2005]. It is a nice property but obviously not a terminating local consistency. To enforce termination, two methods have been tried and combined:

- by forbidding any extraction that would lower some unary function cost. Termination is guaranteed by the fact that these costs can only increase. This has lead to a working definition of AC.
- by imposing an order on variables and allowing extraction of cost from unary function costs only in one direction. This has lead to Directional AC.

Merging both gives Full directional AC.

Consider the weighted network in Figure 1.7(a) using an upper bound  $k = 4$  (the set of costs is  $[0, \dots, 4]$ , with  $\perp = 0$  and  $\top = 4$ ). It has three variables  $X = \{x, y, z\}$  with values  $a, b$ . There are 2 binary constraints  $f_{xz}, f_{yz}$  and two non trivial unary constraints  $f_x$  and  $f_z$ . One optimal solution is eg.  $x = y = z = b$ , with cost 2.

**Node consistency** Consider variable  $z$ . We can project the unary constraint  $f_z$  on the empty scope. We get a non trivial constraint that has constant cost 1. Thus we can replace the initial  $f_\emptyset$  equal to 0 with a new non trivial constraint  $f_\emptyset \oplus f_z[\emptyset]$ . To compensate for this, we also extract it from its source and replace  $f_z$  by  $f_z \ominus f_z[\emptyset]$ . We just enforced node consistency and got an equivalent network 1.7(b). The network has a better obvious lower bound  $f_\emptyset$ . It is node consistent [Larrosa, 2002].

**DEFINITION 7 (NODE CONSISTENCY)** Value  $(i, a)$  is node consistent (NC) if  $f_\emptyset \oplus f_i(a) < k = \top$ . Variable  $i$  is NC\* if:  $(i)$  all its values are NC and  $(ii)$  there exists a value  $a \in D_i$  such that  $C_i(a) = 0$ . Value  $a$  is a support for the variable  $i$ .  $P$  is NC if every variable is NC.

Node consistency characterizes a network where all unary information has been transmitted at the 0-ary level  $(ii)$  and all available information at the 0-ary level that could be transmitted back to the unary level w/o loss of information at the 0-ary level has effectively been transmitted  $(i)$ .

Consider now the remaining network, if we project out variable  $z$  from the cost function  $f_{yz}$ , we get a non trivial unary cost function on  $y$  which is equal to 1 on value  $a$ . We can replace  $f_y$  with  $f_y \oplus f_{yz}[y]$  and replace  $f_{yz}$  with  $f_{yz} \ominus f_{yz}[y]$ . We just enforced a bit of arc consistency and got an equivalent network (c) with a better unary constraint. We can repeat this using  $f_{xz}[z]$  and get another network (d) still more explicit. This network is arc consistent [Schiex, 2000, Larrosa, 2002, Cooper and Schiex, 2004].

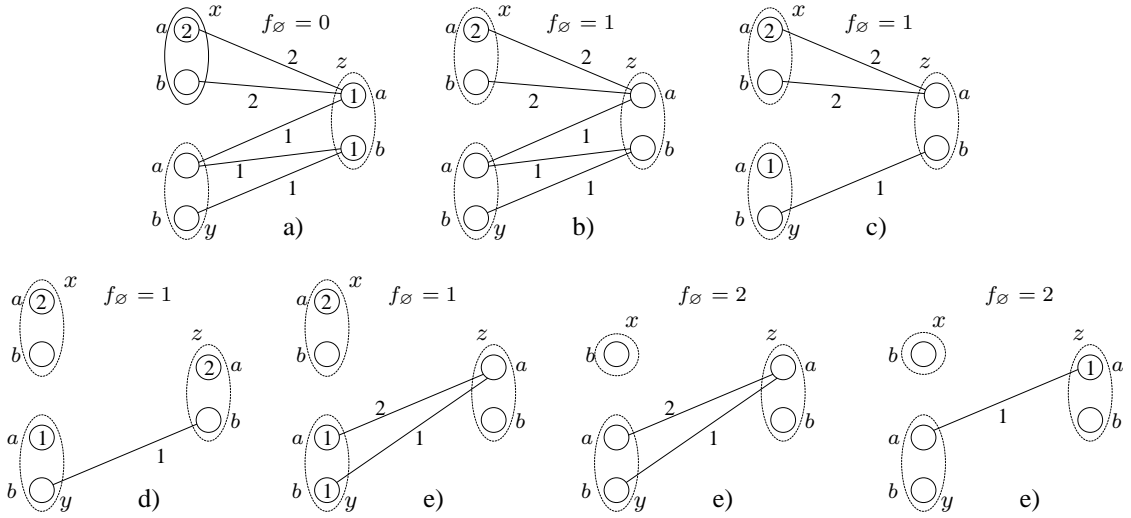


Figure 1.7: A simple weighted CN with  $k = 4$

**DEFINITION 8 (ARC CONSISTENCY)** *Value*  $(i, a)$  *is arc consistent (AC) with respect to constraint*  $f_{ij}$  *if there is a value*  $b \in D_j$  *such that*  $C_{ij}(a, b) = 0$ . *Value*  $b$  *is called a support of the value*  $(i, a)$ . *Variable*  $i$  *is AC if all its values are AC wrt. every binary constraint affecting*  $i$ . *P is AC if every variable is AC and NC.*

Consider now the cost functions  $f_{yz}$  and  $f_z$  together. If we project  $f_{yz} \oplus f_z$  to  $y$  we get a non trivial constraint on  $y$  that maps  $b$  to 1. So we can, 1) replace  $f_{yz}$  by  $f_{yz} \oplus f_z$ , replace  $f_z$  by  $f_z \ominus f_z$  and replace  $f_y$  by  $f_y \oplus (f_{yz} \oplus f_z)[z]$ . This gives the network e) which is not node consistent on  $y$ . We project and extract from  $f_y$  to  $f_\emptyset$ . The lower bound is again increased. Note that value  $a$  of  $x$  can also be deleted (back-propagation from cost to values) because  $f_x(a) \oplus f_\emptyset = 4 = \top$ . This gives the network f).

Here we have combined the information of one unary constraint  $f_z$  together with one binary constraint  $f_{yz}$  to another unary constraint. This more powerful propagation is called directional arc consistency [Cooper, 2003, Larrosa and Schiex, 2003, Cooper and Schiex, 2004]. It assumes that variables are ordered (to avoid termination problems as seen above). Here we assume the order  $w, y, z$ .

**DEFINITION 9 (DIRECTIONAL ARC CONSISTENCY)** *Value*  $(i, a)$  *is directional arc consistent (DAC) wrt. constraint*  $f_{ij}, j > i$ , *if there is a value*  $b \in D_j$  *such that*  $f_{ij}(a, b) \oplus f_j(b) = 0$ . *Value*  $b$  *is called a full support of*  $a$ . *Variable*  $i$  *is DAC if all its values are DAC with respect to every*  $f_{ij}, j > i$ . *P is DAC if every variable is DAC and NC.*

However, our last problem is DAC but not AC because some costs were transferred from  $f_z$  to  $f_{yz}$  but can be brought back to  $f_z$  by projection and extraction from  $f_{yz}$ . We get a problem which is equivalent and both DAC and AC. It said to be fully directional AC [Cooper, 2003, Larrosa and Schiex, 2003, Cooper and Schiex, 2004].

**EXERCICE 26** *Consider a binary weighted CN using a given order of variables. Give an algorithm to enforce DAC on this network along with its spatial and temporal complexities. Imagine that the same network is processed by mini-buckets of size 2. How do the lower bounds induced by each approach compare? Is there any advantage to either approach?*

**DEFINITION 10 (FULL DIRECTIONAL ARC CONSISTENCY)** *P is fully directional arc consistent (FDAC) if it is DAC and AC.*

It is amazing to see that if  $\top = k = 1$  (classical case), AC, and FDAC are equivalent to classical AC, while DAC is equivalent to classical DAC.

Algorithms for enforcing these properties all work using the same schema: they identify one variable that violate the property and enforce the property in this case using the operations outlined in the example (combining projection, combination and extraction). Practical en theoretical efficiency is reached using dedicated data-structures, as in the classical case (ag. in the AC 2001 algorithm). See [Larrosa, 2002, de Givry et al., 2003, Larrosa and Schiex, 2003, Simon de Givry et al., 2005] for details. Algorithms have all polynomial time complexities on binary networks:

- NC:  $O(nd)$
- AC:  $O(n^2d^3)$   $AC > NC$
- DAC:  $O(ed^2)$   $DAC > NC$
- FDAC:  $O(end^3)$   $FDAC > AC, FDAC > DAC$

Spatial complexity is kept in  $O(ed)$  using a simple trick (see [Cooper and Schiex, 2004]). This trick allows to keep the complexity linear in  $d$  on large arity cost functions (represented as analytical formulae for example). See exercise below.

**EXERCICE 27** Consider the extraction of several unary cost functions  $f_{i_1}, \dots, f_{i_m}$  from a cost function of scope  $S, |S| = r$  and all  $x_{i_k} \in S$ . What are the temporal and spatial complexities of a such a sequence of operations? Is it possible to get an algorithm with a spatial complexity which is only linear in  $r$  and which yet gives efficient access to the modified cost function  $f_S \ominus_{j=1}^m f_{i_j}$ ? Is it possible for arbitrary fair valued?

**EXERCICE 28** One usual property of local consistencies is that they deliver a unique result, independently of the order of the enforcing operations. Give a small weighted network that shows that this property is not true anymore for weighted CN and arc consistency as defined above.

Still stronger forms have been defined: cyclic consistency [Martin Cooper, 2004] works on all triangle of constraints on a triplet of variables. It extract from the triangle combination all that can be extracted to  $f_\emptyset$  without creating ternary constraints. This is untested in practice. Introduced more recently, existential arc consistency (EAC [Simon de Givry et al., 2005]) does a similar work on all stars (using all binary constraints involving a common variable). In practice, the local consistency enforced is EAC+DAC+AC, called EDAC.

**EXERCICE 29** It is interesting to compare simple soft arc consistency as defined above with hyper arc-consistency enforced on the classical CN model for soft constraints proposed by Petit et al. [2000].

Propose a small weighted constraint network with an associated upper bound  $k$  such that enforcing hyper arc consistency on the reified version presented in section 1.2.4.2 cannot detect infeasibility but arc consistency as defined above on the original network can. Propose an additional exercise to conclude the comparison.

**EXERCICE 30** The variable elimination algorithm follows a combine/project/forget algorithm. Is it possible to build a combine/project/extract version of variable elimination? Compare with the original version in terms of services offered, spatial and temporal complexity.

**EXERCICE 31** SAT being a specific case of CSP, weighted MaxSAT is a specific case of weighted CN. Consider two weighted 2-clauses with a common variable but with a different sign. Apply one of the above local consistency to this and give the resulting set of clauses. Compare with resolution in classical logic (see [Javier Larrosa and Federico Heras, 2005]). Propositional logic with weights is also called “penalty logic” [Pinkas, 1991, Dupin de Saint Cyr et al., 1994] and is used for non-monotonic reasoning.

**Notes** The first contribution in this area extended arc consistency, the most usual local consistency property, to fuzzy constraint networks [Rosenfeld et al., 1976]. The property and associated polynomial time algorithm proposed in this paper have been refined and reformulated in [Snow and Freuder, 1990, Schiex, 1992]. We remind the reader of the very strong relation between fuzzy constraint networks and classical constraint networks which shows that the extension to the fuzzy networks case can always be done with an  $O(\log(l))$  multiplication in complexity (where  $l$  is the number of levels used in the network).

#### 1.4.2.4 Soft global constraints

One important class of consistency enforcing algorithms in classical networks is the class of so-called global constraints (see the associated notes on global constraints by J-C. Rgin). Several usual global constraints and their associated algorithms have been extended to handle soft constraints. All these proposals have been made using the approach of [Petit et al., 2000] where a cost function  $f_S$  is represented as an hard constraint with an extra variable  $x_S$  representing the cost of the assignment of the other variables in  $S$  (see § 1.2.4.2).

A global soft constraint is defined by three components: the precise semantics of the constraints, the level of consistency enforced on this constraint and an algorithm to enforce it. For example, one simple soft global constraint extends the classical all-different constraint. Two semantics have been considered for a soft version: for a given assignment of the variables involved in a soft all-different, the associated level can be either the number of variables whose value must be changed to satisfy the all-different constraint or the number of pairs of variables that have identical values. The level of consistency enforced is classical generalized arc-consistency also called hyper-arc consistency. Enforcing algorithms based on flow/matching algorithms offer efficient enforcing algorithms for these two semantics [Petit et al., 2001, van Hoesve, 2004].

Before this, [Baptiste et al., 1998] first proposed a soft global constraint handling a variant of the One-Machine scheduling problem. Following this first proposal, a few extra soft global constraints have been proposed. Besides the previous soft all-different constraint, soft versions of the global cardinality constraint (useful for example in personnel rostering problems) and of the regular constraint (to capture regular language membership with errors) have also been proposed by [van Hoesve et al., 2004].

The problem of just computing the cost of an assignment for a single soft global constraint has been considered in [Beldiceanu and Petit, 2004]. For some semantics, this problem may naturally be NP-hard but all global constraints defined through specific graph properties can be computed in polynomial time.

#### 1.4.3 Polynomial classes

The previous section on variable elimination shows that all structural classes of classical CN extend immediately to arbitrary valued CN. Actually, many existing works such as [Shenoy, 1991, Bistarelli et al., 1995, Dechter, 1999] show that these structural properties can be exploited in many cases and not only for optimization (but also eg. for discrete integration). So, a semiring CN with a graph which can be covered by a  $k$ -tree can be solved in time polynomial in  $k$ . Specifically, tree-structured CN can be solved efficiently.

*EXERCICE 32 Show that a tree structured weighted CN can be solved optimally by DAC enforcing. What variable order should be used ? Prove that you get at the end a compact representation of all solutions along with their costs.*

Note that there is a huge bibliography on this topic including extensions to hypergraphs.

##### 1.4.3.1 Fuzzy networks

The idempotent valued case (i.e. fuzzy constraint networks) is specific again because of its close relationship with classical CSP. Consider the Exercice 2. Replace the oracle for CSP by a polynomial time oracle for your class. For some polynomial class (eg. CSP with 2 values only), this is a direct extension.

EXERCICE 33 Consider the polynomial class of temporal CN. What condition can be imposed on the cost functions of a fuzzy temporal CN so that the polynomial class extends? Can you lift any other class this way ?

See [Khatib et al., 2001, Rossi et al., 2001, 2002a,b] for more details on the fuzzy TCSP.

### 1.4.3.2 Weighted CN

Because even the simple Max2SAT is NP-complete one may wonder if there is any tractable language for valued constraint network. A tractable language would be a set of functions that any any network that uses only these functions can be solved in polynomial time.

In the simple Max-2SAT case, even the simple language where only exclusive or constraints are used is NP-complete. Tractable languages for MaxSat have been completely characterized by [Creignou, 2001].

The simple MaxCSP problem offers larger possible domains and additional complexity. In the case of Max-CSP, the language of soft binary equality:

$$f_{eq}(x, y) = \begin{cases} 0 & x = y \\ 1 & \text{otherwise} \end{cases}$$

is NP-hard.

EXERCICE 34 Give a polynomial reduction from the MINIMUM 3-TERMINAL CUT.

MIN. 3-TERMINAL CUT: *an undirected (weighted) graph  $G = (V, E)$ . Three distinguished vertices  $\{v_1, v_2, v_3\}$ . Is there a set of edges of minimum weight whose removal disconnects each pair of terminals?*

Amazingly there is a non trivial and not completely unpractical polynomial time language for cost functions (that applies to a subset of fair valued CN, at least strictly idempotent one). We just introduce the notion of generalized interval functions [David Cohen et al., 2004c].

Imagine that domain  $D$  ordered. A binary cost function  $D^2 \rightarrow E$  is a generalized interval function on  $D$  if it has the following form:

$$c_{[a,b]}^\rho(x, y) = \begin{cases} 0 & : (x < a) \vee (y > b) \\ \rho & : \text{otherwise} \end{cases}$$

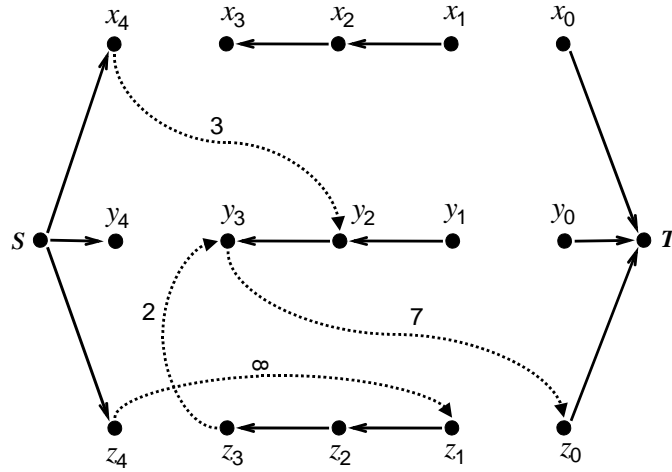
Such functions have cost  $0 = \perp$  everywhere but on a rectangular region in the corner of their cost matrix and are, in their hard version, a specific case of connected row-convex constraints.

It has been shown by [David Cohen et al., 2004c] that the language of GI functions is tractable. The tractability proof is elegant and relies on a nice transformation to a graph problem.

Consider  $P = \langle X, D, C \rangle$  a maxCSP with  $D_i = \{1, \dots, M\}$ . Then build the graph  $G = (V, E)$  with:

- $V = \{S, T\} \cup \{x_{id} \mid x_i \in X, d \in D\{1, \dots, M\}\}$ .
- for each  $x_i \in X$ , an edge from  $S$  to  $x_{iM}$  weight  $\infty$
- for each  $x_i \in X$ , an edge from  $x_{i0}$  to  $T$ , weight  $\infty$
- for each  $x_{id} \in V, d \in [1, M - 2]$ , an edge from  $x_{id}$  to  $x_{id+1}$  with weight  $\infty$ .
- for each constraint  $c_{[a,b]}^\rho(x_i, x_j)$  an edge from  $x_{jb}$  to  $x_{ia-1}$  with weight  $\rho$  (c-edges).

Consider the following problem with variables  $X = \{x, y, z\}$ , having domains all equal to  $\{1, 2, 3, 4\}$  and a set of cost functions  $C = \{c_{[3,4]}^3(y, x), c_{[1,3]}^7(z, y), c_{[4,3]}^2(y, z), c_{[2,4]}^\infty(z, z)\}$ , then the graph obtained is:



DEFINITION 11 A minimal  $S - T$  cut that contains only  $c$ -edges is a proper cut

In the graph, this is  $\{\langle y_3, z_0 \rangle\}, \{\langle x_4, y_2 \rangle, \langle z_3, y_3 \rangle\}$ .

THEOREM 2 For each minimal proper cut of weight  $\Phi$ , there is an assignment of cost  $\Phi$  and vice-versa.

Here:  $\{\langle y_3, z_0 \rangle\}$  has weight 7,  $\{\langle x_4, y_2 \rangle, \langle z_3, y_3 \rangle\}$  has weight 5. Both are minimal.

EXERCICE 35 Prove the theorem.

Using any algorithm for minimum weighted cut therefore solves the problem. This class has been extended using a decomposition of submodular functions to GI functions by the same authors to the general class of submodular functions.

DEFINITION 12 A function such  $\forall x, y, u, v, u \leq x, v \leq y$ , we have:

$$c(u, v) + c(x, y) \leq c(u, y) + c(x, v)$$

is called a submodular function.

A submodular function cost matrix decomposes in a sum of GI functions. This class is relatively rich in practice. Here are examples of such functions:

$$ax + by + c, \sqrt{x^2 + y^2}, |x - y|^r (r \geq 1), \max(x, y, 0)^r (r \geq 1)$$

This class is actually maximal (no other function can be added to the language without making it NP-complete). An (unimplemented) algorithm in  $O(n^3 d^3)$  can solve submodular networks. Several related results appear in [David Cohen et al., 2003, 2004a,c,b].

## 1.5 Combining search and inference

### 1.5.1 Direct combination

As section 1.4 has shown, variable (or bucket) elimination is computationally and space efficient when the variable to eliminate is only connected to few other variables or when it is assigned. Once the variable is eliminated, we get a network with the same optimal cost, less variables and constraints and no backtracking. Conversely, Branch and bound explores the domain of every variable with limited space complexity but with the requirement of backtracking until a provably optimal solution is found. This gives a natural way to combine both approaches: if some variable in the network has small degree, eliminate it. Each elimination may reduce the degree of other variables and enable further eliminations. Otherwise branch on a well-chosen variable. Once assigned, the variable becomes easy to eliminate and may enable further eliminations.

## 1.5.2 Exploiting stronger bounds

### 1.5.2.1 Local consistency based bounds

We have seen that local consistency enforcing, by increasing the value of  $f_\emptyset$ , can provide increased lower bounds for a given problem. We can just inject these lower bounds in the branch and bound algorithm by maintaining some form of local consistency on the subproblem associated with each node. The value of  $f_\emptyset$  after enforcing should give an improved lower bound.

Maintaining AC on fuzzy networks has been first considered in [Schiex, 1992]. For weighted network, more choices are now possible: the level of local consistency enforced (NC, AC, DAC, FDAC, EDAC = EAC+FDAC) this gives various quality of lower bounds. See [Larrosa, 2002, Larrosa and Schiex, 2003, Simon de Givry et al., 2005]. In

All these local consistencies have been implemented and are part of an efficient experimental tool applicable to weighted CN and weighted SAT called `toolbar` (see section 1.6.1). In the current situation, the strongest (EDAC) local consistency seems to provide the best results, especially on hard problems.

### 1.5.2.2 Mini-buckets based bounds

Similarly, the bounds induced by mini-buckets can be injected inside a tree-search algorithm. However, computing mini-buckets is exponential in the mini-bucket parameter. To avoid repeated evaluation of the lower bound induced by mini-buckets on the subproblem associated with each node of the search tree, and similarly to what was done in the Russian Dolls Search algorithm, it is possible to avoid this by assuming that a fixed variable ordering is used (this is a very costly assumption in practice).

Consider a valued CN  $\langle X, D, C, S \rangle$  and assume the variable ordering  $x_1, \dots, x_n$  is used to assign variables in a Branch and bound search. Before starting branch and bound we process the soft network using mini-buckets. We process each variable in an order opposite to the assignment order. When processing variable  $x_j$ , we will build a family of cost functions of bounded maximum arity  $f_i^{x_j}$ . Note that by definition:

- $f_i^{x_j}$  can only involve variables before  $x_j$  in the assignment ordering.
- $f_i^{x_j}$  has been built by combining together functions whose scope contains only variables which are after  $x_j$  (including it) in the assignment order.

The extra functions are not included in the original problem but memorized for later use during branch and bound.

Consider now a node of the search, assigning variables  $x_1$  to  $x_p$ . We now for sure that the  $f_\emptyset$  produced by assigning cost functions is a lower bound that is obtained using only cost functions whose scope is included in  $\{x_1, \dots, x_p\}$ . Consider now the extra  $f_i^{x_j}$  produced by mini-buckets such that 1)  $x_j > x_p$  in the assignment order and 2) their scope is included in  $\{x_1, \dots, x_p\}$ .

Because of 2) we can easily compute their value since the variables in their scope are assigned. Because of 1) we know that they form a lower bound which is independent of  $f_\emptyset$  and thus can be combined with it. This defines a new lower bound which is better than  $f_\emptyset$  alone and a branch and bound algorithm denoted as  $BMB(i)$  where  $i$  is the parameter that bounds the maximum number of variables used together to produce the mini-buckets.

The larger  $i$ , the stronger the bound will be but the more expensive to compute it is. Finding the best or even a good  $i$  is important and not obvious (no heuristics procedure for this are known).

This scheme was first presented in [Kalev Kask and Rina Dechter, 1999] and has later been enhanced in [Dechter et al., 2001]. Algorithms based on local consistency maintenance seem still quite competitive and do not require any parameter tuning.

## 1.6 Using soft constraints

### 1.6.1 Existing solvers and resources

Many classical CN solvers include optimization primitives that allows a first approach to soft constraints using the approach of [Petit et al., 2000]. However, and except for specific cases such as described in [T. Benoist and M. Lemaître, 2003], enforcing AC on this representation is weaker than working on non reified soft constraints directly using local consistencies.

A large subset of the algorithms presented in this document (including bucket elimination, maintaining AC, DAC, FDAC and EDAC) is available in the open source collaboratively developed C solver `toolbar`. It is capable of reading weighted CN and weighted SAT (CNF) problems. Bayesian nets support for MPE (Maximum Probability Explanation) will soon be available. From our current knowledge, it seems to be among the fastest general weighted MaxCSP/MaxSAT solver available. It can be downloaded at <http://carlit.toulouse.inra.fr/cgi-bin/awki.cgi/ToolBarIntro>. A similar tool is developed in the laboratory of R. Dechter by R. Marinescu (you have to contact Radu for getting the C++ sources but Windows binaries are available on <http://www.ics.uci.edu/~radum/>).

But many other solvers have also been built in the last decades, including local search solvers. A Wiki web site of cost function optimizers is available at <http://carlit.toulouse.inra.fr/cgi-bin/awki.cgi/Algorithms>. If your solver is not there, please add it (the password is `bia31`). There are also many pure MaxSAT and pseudo boolean solvers for SAT which have been made available recently. See [de Givry et al., 2003, Hantao Zhang et al., 2003, H. Shen and H. Zhang, 2004, Zhao Xing and Weixiong Zhang, 2005, Simon de Givry et al., 2005, Javier Larrosa and Federico Heras, 2005] for example.

See also:

- *Con'Flex*: Conjunctive fuzzy CSP system with integer, symbolic and numerical constraints ([www.inra.fr/bia/T/conflex](http://www.inra.fr/bia/T/conflex)).
- *clp(FD,S)*: semi-ring CLP ([pauillac.inria.fr/~georget/clp\\_fds/clp\\_fds.html](http://pauillac.inria.fr/~georget/clp_fds/clp_fds.html)).
- *LVCSP*: Common-Lisp library for Valued CSP with an emphasis on strictly monotonic operators ([ftp.cert.fr/pub/lemaitre/LVCSP](http://ftp.cert.fr/pub/lemaitre/LVCSP)).
- *Choco*: a claire library for CSP. Existing layers above Choco implements Weighted Max-CSP algorithms ([choco.sourceforge.net](http://choco.sourceforge.net)).

(although some of these links may be a bit outdated now).

A reasonable large collection of benchmarks for weighted CN and weighted SAT has been built over the last years and is available at <http://carlit.toulouse.inra.fr/cgi-bin/awki.cgi/BenchmarkS> in a standardized format. This format is read directly by some other solvers including local search solvers such as `incop`.

### 1.6.2 Some applications

Several problems can be directly translated in soft CN. First many usual and central problems in complexity theory such MaxSAT, MaxOnes, Min Vertex Cover, MaxCut, Min-Ones, MinCOL... These academic problems have sometimes almost direct applications in various areas (eg. Min Vertex Cover is related to two-level logic minimization in electronic design automation, MinCol is a simplified version of overconstrained frequency assignment...).

Many real problem such as personal rostering, timetabling, frequency assignment problems, satellite scheduling, RNA gene finding, pedigree error detection are better represented using soft CN. Some problems modelled using soft constraints are described in the following papers [Cabon et al., 1999, Bensana et al., 1999, Gaspin, 2001, Bistarelli et al., 2001, Steven Prestwiteh et al., 2003, Samir Loudni and Patrice Boizumault, 2003, John Slaney et al., 2004, de Givry et al., 2005, Gaspin et al., 2005].

Soft constraints can be used at different levels: for modelling the problem, using soft constraints as guide to find good solutions rapidly, for directly solving problem modeled.

I just present here one famous application the area, the frequency assignment problem defined by the CELAR (Centre d'électronique de l'Armement). This problem is defined in detail in Cabon et al. [1999] and an excellent web site is <http://fap.zib.de> where the different flavors of the problem are presented.

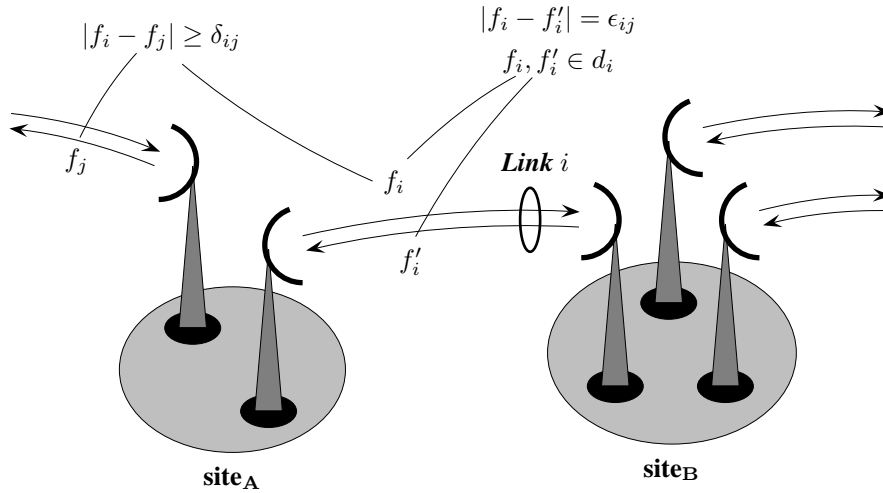


Figure 1.8: Frequency assignment

### 1.6.3 Frequency assignment

A set of wireless communication connections must be assigned frequencies such that for every connection data transmission between the transmitter and receiver is possible. The frequencies should be selected from a given set that may depend on the location (in practice, much traffic is bidirectional, so that in fact two frequencies must be chosen, one for each direction but this is often ignored by choosing two non intersecting domains of frequency for forward and backward communication).

The frequencies assigned to two connections may incur interference resulting in a loss of quality of the signal. Two conditions must be fulfilled in order to have interference of two signals:

- The two frequencies must be close on the electromagnetic band (Doppler effects).
- The connections must be geographically close to each other. The signals that may interfere should have a similar level of energy at the position where they might disturb each other.

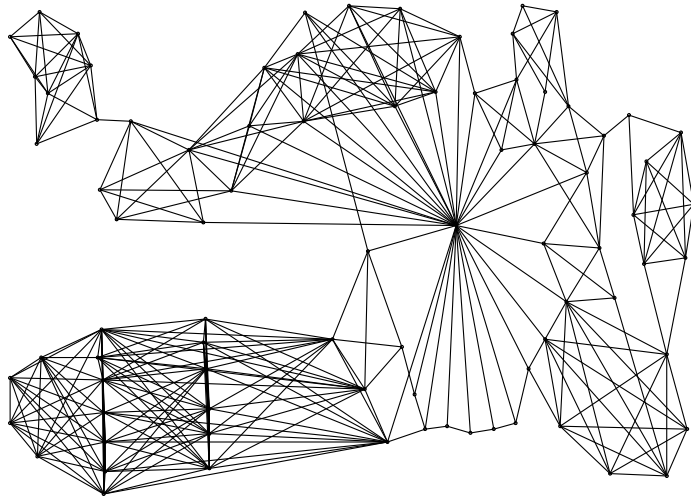
When the second condition is satisfied, and depending on an physical wave propagation model, a sufficient distance in the frequency spectral has to be imposed. Because the frequency resource is not infinite, some frequencies have to be reallocated and the problem of finding an assignment that satisfies all distance constraints is already NP-hard.

Often, one also want to minimize some criteria. Minimizing the maximum frequency used allows to use a small portion of an available spectrum. Minimizing the number of different frequencies used allows to rapidly find an available frequency for a new link. When no solution exists that satisfy all distance constraint, the criteria is usually changed to minimize interference. In the CELAR case, the aim is to minimize a weighted sum of violated distance constraints.

The problem is easy to model using one variable per link whose domain is the set of available frequencies for the link. Constraints of the form  $|f_i - f_j| > \epsilon_{ij}$  are used to specify the minimum distance between geographically close links. If the problem is overconstrained, these hard constraints are made soft: a satisfactory assignment has  $\perp = 0$  cost and otherwise a fixed  $p_{ij}$  cost. The aim is then to minimize the sum of all costs emitted which is a clear instance of weighted MaxCSP.

Note that minimizing the maximum frequency used is a min-max fuzzy constraint network using only unary soft constraints. Minimizing the number of frequency used is best modeled using a soft global constraint.

The problems have been tackled using many different combinatorial optimization techniques over time. Several are still open. All min-max problem have been solved using constraint network technology. The



---

Figure 1.9: Frequency assignment graph structure of CELAR instance 6

first overconstrained instance has been closed using a combination of graph partitioning and Russian Dolls Search Cabon et al. [1999]. The graph of the corresponding instance is visible in the Figure 1.9. This very specific structure is an excellent support to dynamic programming (variable elimination) algorithms and other instances have been later closed using such techniques in Koster [1999].

## Conclusion

Soft constraint technology has made incredible progress in the last decade. Even if there are still many theoretical and algorithmic progress to do (still stronger lower bounds seems needed), we do have enough to start building solid soft constraint systems. The `toolbar` system is one step in this direction. One more compulsory step is a form of integration of the algorithms and properties developed here and available classical constraint technology. This integration is in some sense already done in the “constrained optimization” approach of Petit et al. [2000] but the integration of recent algorithms is not easy nor elegant in this framework. More integrative solutions have to be found.

To deal with many more practical problems, several issues will have to be addressed in this framework (large domains...).

# Bibliography

- E. Aarts and J. K. Lenstra. *Local Search in Combinatorial Optimization*. Interscience Series in Discrete Mathematics and Optimization. John Wiley and sons, 1997.
- Adnan Darwiche. Recursive Conditioning. *Artificial Intelligence*, 126(1-2):5–41, 2001.
- M. S. Affane and H. Bennaceur. A weighted arc consistency technique for Max-CSP. In *Proc. of the 13<sup>th</sup> ECAI*, pages 209–213, Brighton, United Kingdom, 1998.
- P. Baptiste, C. L. Pape, and L. Peridy. Global constraints for partial CSPs: a case-study of resource and due date constraints. In *Proc. 4th Int. Conf. on Principles and Practice of Constraint Programming (CP98)*. Springer-Verlag, LNCS 1520, 1998.
- N. Beldiceanu and T. Petit. Cost Evaluation of Soft Global Constraints. In *Proc. of CP-AI-OR'2004 international conference*, pages 80–95, Nice, France, 2004.
- E. Bensana, M. Lemaître, and G. Verfaillie. Earth observation satellite management. *Constraints*, 4(3): 293–299, 1999.
- U. Bertelé and F. Brioshi. *Nonserial Dynamic Programming*. Academic Press, 1972.
- S. Bistarelli, U. Montanari, and F. Rossi. Constraint solving over semirings. In *Proc. of the 14<sup>th</sup> IJCAI*, Montréal, Canada, Aug. 1995.
- S. Bistarelli, U. Montanari, and F. Rossi. Semiring based constraint solving and optimization. *Journal of the ACM*, 44(2):201–236, 1997.
- S. Bistarelli, U. Montanari, and F. Rossi. Semiring-based constraint logic programming: Syntax and semantics. *ACMTOPLAS: ACM Transactions on Programming Languages and Systems*, 23, 2001.
- A. Borning, M. Mahert, A. Martindale, and M. Wilson. Constraint hierarchies and logic programming. In *Int. conf. on logic programming*, pages 149–164, 1989.
- C. Beeri, R. Fagin, D. Maier, and M. Yannakakis. On the desirability of acyclic database schemes. *Journal of the ACM*, 30:479–513, 1983.
- B. Cabon, S. de Givry, L. Lobjois, T. Schiex, and J. Warners. Radio link frequency assignment. *Constraints Journal*, 4:79–89, 1999.
- Cédric Pralet and Gérard Verfaillie. Travelling in the World of Local Searches in the Space of Partial Assignments. In *Proceedings of the CPAIOR 2004 international conference*, pages 240–255, 2004.
- Christian Blum and Andrea Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Comput. Surv.*, 35(3):268–308, 2003.
- M. Cooper and T. Schiex. Arc consistency for soft constraints. *Artificial Intelligence*, 154(1-2):199–227, Apr. 2004. see [arXiv.org/abs/cs.AI/0111038](http://arXiv.org/abs/cs.AI/0111038).
- M. C. Cooper. Reduction operations in fuzzy or valued constraint satisfaction. *Fuzzy Sets and Systems*, 134(3):311–342, 2003.

- K. Creignou, N. *Complexity Classifications of Boolean Constraint Satisfaction Problems*. Vol. 7 of SIAM Monographs on Discrete Mathematics and Applications, 2001.
- Cyril Terrioux and Philippe Jégou. Bounded backtracking for the valued constraint satisfaction problems. *Proceedings of the Ninth International Conference on Principles and Practice of Constraint Programming CP'03*, pages 709–723, 2003.
- David Cohen, Martin Cooper, and Peter Jeavons. A complete characterization of complexity for Boolean constraint optimization problems. In *Proceedings of CP'04*, number 3258 in LNCS, pages 212–226, 2004a.
- David Cohen, Martin Cooper, Peter Jeavons, and Andrei A. Krokhin. Identifying efficiently solvable cases of Max CSP. In *Proceedings of STACS'04*, number 2996 in LNCS, pages 152–163, 2004b.
- David Cohen, Martin Cooper, Peter Jeavons, and Andrei Krokhin. Soft constraints: complexity and multimorphisms. In *Proceedings of CP'03*, number 2833 in LNCS, pages 244–258, 2003.
- David Cohen, Martin Cooper, Peter Jeavons, and Andrei Krokhin. A maximal tractable class of soft constraints. *Journal of Artificial Intelligence Research*, 22:1–22, 2004c.
- S. de Givry and L. Jeannin. A unified framework for partial and hybrid search methods in constraint programming. *Computer & Operations Research*, 2004.
- S. de Givry, J. Larrosa, P. Meseguer, and T. Schiex. Solving Max-Sat as weighted CSP. In *Proc. of the Ninth International Conference on Principles and Practice of Constraint Programming*, LNCS, Kinsale, Ireland, Oct. 2003. Springer Verlag.
- S. de Givry, Z. Vitezica, and T. Schiex. Mendelian error detection in complex pedigree using weighted constraint satisfaction techniques. In *Proc. of the Workshop on Constraint Based Methods for Bioinformatics*, Sitges, Spain, 2005.
- R. Dechter. Mini-buckets: A general scheme for generating approximations in automated reasoning. In *IJCAI*, pages 1297–1303, 1997.
- R. Dechter. Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence*, 113(1–2): 41–85, 1999.
- R. Dechter, K. Kask, and J. Larrosa. A general scheme for multiple lower bound computation in constraint optimization. In *Principles and Practice of Constraint Programming - CP 2001*, volume 2239 of LNCS, pages 346–360, Paphos, Cyprus, Nov. 2001. Springer Verlag.
- R. Dechter and J. Pearl. Tree clustering for constraint networks. *Artificial Intelligence*, 38:353–366, 1989.
- F. Dupin de Saint Cyr, J. Lang, and T. Schiex. Penalty logic and its link with dempster-shafer theory. In *Proc. of the 10<sup>th</sup> International Conference on Uncertainty in Artificial Intelligence*, 1994.
- K. Evans, M. Konikoff, R. Mathis, J. Maden, and G. Whipple. Totally ordered commutative monoids. *Semigroup Forum*, 62(2):249–278, 2001.
- H. Fargier and J. Lang. Uncertainty in constraint satisfaction problems: a probabilistic approach. In *Proc. of ECSQARU '93*, LNCS 747, pages 97–104, Granada, Spain, Nov. 1993.
- E. Freuder and R. Wallace. Partial constraint satisfaction. *Artificial Intelligence*, 58:21–70, Dec. 1992.
- E. C. Freuder. Partial constraint satisfaction. In *Proc. of the 11<sup>th</sup> IJCAI*, pages 278–283, Detroit, MI, 1989.
- E. C. Freuder and M. J. Quinn. Taking advantage of stable sets of variables in constraint satisfaction problems. In *Proc. of the 9<sup>th</sup> IJCAI*, pages 1076–1078, Los Angeles, CA, 1985.
- P. Galinier and J.-K. Hao. Tabu search for maximal constraint satisfaction problems. In *Principles and Practice of Constraint Programming - CP'97*, number 1330 in LNCS, pages 196–208, 1997.

- C. Gaspin. RNA Secondary Structure Determination and Representation Based on Constraints Satisfaction. *Constraints*, 6(2-3):201–221, June 2001.
- C. Gaspin, S. de Givry, T. Schiex, P. Thbault, and M. Zytnicki. A new local consistency for weighted csp applied to ncna detection. In *Proc. of the Workshop on Constraint Based Methods for Bioinformatics*, Sitges, Spain, 2005.
- H. Shen and H. Zhang. Study of lower bound functions for MAX-2-SAT. In *Proc. of 19th National Conference on Artificial Intelligence (AAAI'04)*, San Jose, California, July 2004.
- Hantao Zhang, Haiou Shen, and Felip Manyà. Exact Algorithms for MAX-SAT. *Electr. Notes Theor. Comput. Sci.*, 86(1), 2003.
- J. Larrosa and E. Morancho. Solving 'Still life' with soft constraints and bucket elimination. In *Proc. of Principles and Practice of Constraint Programming (CP'03)*, Kinsale, Cork County, Ireland., 2003.
- Javier Larrosa and Federico Heras. Resolution in Max-SAT and its relation to local consistency in weighted CSPs. In *Proc. of the 19<sup>th</sup> IJCAI*, Edinburgh, Scotland, 2005.
- Javier Larrosa, Pedro Meseguer, and Martí Sánchez. Pseudo-tree Search with Soft Constraints. In *Proceedings of the 15th European Conference on Artificial Intelligence*, pages 131–135, 2002.
- John Slaney, Arnold Binas, and David Price. Guiding a Theorem Prover with Soft Constraints. In *Proc. of ECAI'04*, 2004.
- Kalev Kask and Rina Dechter. Branch and Bound with Mini-Bucket Heuristics. In *Proc. of the 16<sup>th</sup> IJCAI*, pages 426–443, Stockholm, Sweden, 1999.
- L. Khatib, P. Morris, R. Morris, and F. Rossi. Temporal constraint reasoning with preferences. In *Proc. of the 17<sup>th</sup> IJCAI*, pages 322–327, Washington, USA, Aug. 2001.
- E. Klement, R. Mesiar, and E. Pap. *Triangular Norms*. Kluwer academic Publishers, 2000.
- A. M. Koster. *Frequency assignment: Models and Algorithms*. PhD thesis, University of Maastricht, The Netherlands, Nov. 1999. Available at [www.zib.de/koster/thesis.html](http://www.zib.de/koster/thesis.html).
- J. Larrosa. On arc and node consistency in weighted csp. In *Proc. AAAI'02*, Edmondton, (CA), 2002.
- J. Larrosa and P. Meseguer. Phase transition in max-csp. In *Proc. of ECAI'96*, pages 190–194, Budapest, Hungary, 1996.
- J. Larrosa and P. Meseguer. Partition-based lower bound for max-csp. In *Proc. of the 5<sup>th</sup> International Conference on Principles and Practice of Constraint Programming (CP-99)*, pages 303–315, 1999.
- J. Larrosa and T. Schiex. In the quest of the best form of local consistency for weighted CSP. In *Proc. of the 18<sup>th</sup> IJCAI*, pages 239–244, Acapulco, Mexico, Aug. 2003.
- T. Lau and P. Tsang. Guided genetic algorithm and its application to radio link frequency assignment problems. *Constraints*, 6(4):373–398, 2001.
- Laurent Michel and P. Van Hentenryck. Localizer. *Constraints*, 5:41–82, 2000.
- Martin Cooper. Cyclic consistency: a local reduction operation for binary valued constraints. *Artificial Intelligence*, 155(1-2):69–92, May 2004.
- Martin Cooper. Hig-Order Consistency in Valued Constraint Satisfaction. *Constraints*, 10:283–305, 2005.
- P. Meseguer, J. Larrosa, and M. Sanchez. Lower bounds for non-binary constraint optimization problems. In *Principles and Practice of Constraint Programming - CP 2001*, volume 2239 of *LNCS*, pages 317–331, Paphos, Cyprus, Nov. 2001. Springer Verlag.

- P. Meseguer and M. Sanchez. Specializing russian doll search. In *Principles and Practice of Constraint Programming - CP 2001*, volume 2239 of *LNCS*, pages 464–478, Paphos, Cyprus, Nov. 2001. Springer Verlag.
- P. Meseguer, N. Bouhmala, T. Bouzoubaa, M. Irgens, and M. Sánchez. Current approaches for solving overconstrained problems. *Constraints*, 8(1):9–39, 2003.
- C. M. Papadimitriou. *Computational Complexity*. Addison-Wesley Publishing Company, 1994. ISBN 0-201-53082-1.
- Pedro Meseguer, Marti Sánchez, and Gérard Verfaillie. Opportunistic Specialization in Russian Doll Search. In *Proceedings of Principles and Practice of Constraint Programming CP'02*, volume 2470 of *LNCS*, pages 264–279, 2002.
- T. Petit, J.-C. Régin, and C. Bessière. Specific filtering algorithms for over-constrained problems. In *Principles and Practice of Constraint Programming - CP 2001*, volume 2239 of *LNCS*, pages 451–463, Paphos, Cyprus, Nov. 2001. Springer Verlag.
- T. Petit, J.-C. Rgin, and C. Bessire. Meta-constraints on violations for over constrained problems. In *IEEE-ICTAI'2000 international conference*, pages 358–365, Vancouver, Canada, Nov. 2000.
- G. Pinkas. Propositional non-monotonic reasoning and inconsistency in symetric neural networks. In *Proc. of the 12<sup>th</sup> IJCAI*, pages 525–530, Sidney, Australia, 1991.
- Radu Marinescu and Rina Dechter. AND/OR Branch-and-Bound for Graphical Models. In *Proc. of the 19<sup>th</sup> IJCAI*, Edinburgh, Scotland, 2005.
- J.-C. Régin, T. Petit, C. Bessière, and J.-F. Puget. New lower bounds of constraint violations for over-constrained problems. In *Principles and Practice of Constraint Programming - CP 2001*, volume 2239 of *LNCS*, pages 332–345, Paphos, Cyprus, Nov. 2001. Springer Verlag.
- A. Rosenfeld, R. Hummel, and S. Zucker. Scene labeling by relaxation operations. *IEEE Trans. on Systems, Man, and Cybernetics*, 6(6):173–184, 1976.
- F. Rossi, A. Sperduti, L. Khatib, P. Morris, and R. A. Morris. Learning preferences on temporal constraints: a preliminary report. In *TIME*, pages 63–68, 2001.
- F. Rossi, A. Sperduti, K. Venable, L. Khatib, P. Morris, and R. Morris. Learning and solving soft temporal constraints: An experimental study. In *Proc. CP 2002*, LNCS, Ithaca, NY, Sept. 2002a. Springer Verlag.
- F. Rossi, K. Venable, L. Khatib, P. Morris, and R. Morris. Two solvers for tractable temporal constraints with preferences. In *Proc. AAAI 2002 workshop on preference in AI and CP*, Edmonton, Canada, July 2002b.
- Z. Ruttkay. Fuzzy constraint satisfaction. In *Proc. FUZZ-IEEE'94*, Orlando, Florida, 1994.
- Samir Loudni and Patrice Boizumault. Solving Constraint Optimization Problems in Anytime Contexts. In *Proc. of the 18<sup>th</sup> IJCAI*, pages 251–256, Acapulco, Mexico, 2003.
- T. Schiex. Possibilistic constraint satisfaction problems or “How to handle soft constraints?”. In *Proc. of the 8<sup>th</sup> Int. Conf. on Uncertainty in Artificial Intelligence*, Stanford, CA, July 1992.
- T. Schiex. Arc consistency for soft constraints. In *Principles and Practice of Constraint Programming - CP 2000*, volume 1894 of *LNCS*, pages 411–424, Singapore, Sept. 2000.
- T. Schiex, H. Fargier, and G. Verfaillie. Valued constraint satisfaction problems: hard and easy problems. In *Proc. of the 14<sup>th</sup> IJCAI*, pages 631–637, Montréal, Canada, Aug. 1995.
- L. Shapiro and R. Haralick. Structural descriptions and inexact matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3:504–519, 1981.

- P. Shenoy. Valuation-based systems for discrete optimization. In Bonissone, Henrion, Kanal, and Lemmer, editors, *Uncertainty in AI*. North-Holland Publishers, 1991.
- Simon de Givry, Federico Heras, Javier Larrosa, and Matthias Zytnicki. Existential arc consistency: getting closer to full arc consistency in weighted CSPs. In *Proc. of the 19<sup>th</sup> IJCAI*, Edinburgh, Scotland, 2005.
- S.L. Lauritzen and D.J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society – Series B*, 50:157–224, 1988.
- P. Snow and E. Freuder. Improved relaxation and search methods for approximate constraint satisfaction with a maximin criterion. In *Proc. of the 8<sup>th</sup> biennial conf. of the canadian society for comput. studies of intelligence*, pages 227–230, May 1990.
- Steven Prestwich, Des Higgins, and Orla O’Sullivan. A SAT-Based Approach to Multiple Sequence Alignment. In *Proc. of Principles and Practice of Constraint Programming (CP’03)*, number 2833 in LNCS, pages 940–944, 2003.
- T. Benoist and M. Lemaître. An elegant and efficient implementation of russian dolls search for variable weighted CSP. In *Proceedings of the 5th International Workshop on Soft Constraints*, Kinsale, Ireland, 2003.
- W. J. van Hoesve. A Hyper-Arc Consistency Algorithm for the soft AllDifferent Constraint. In *Proc. of the tenth international conference on Principles and Practice of Constraint Programming (CP 2004)*, number 3258 in LNCS, 2004.
- W. J. van Hoesve, G. Pesant, and L.-M. Rousseau. On Global Warming (Softening Global Constraints). In *Proc. of the 6th International Workshop on Preferences and Soft Constraints*, Toronto, Canada, 2004.
- G. Verfaillie, M. Lemaître, and T. Schiex. Russian doll search. In *Proc. of AAAI’96*, pages 181–187, Portland, OR, 1996.
- R. Wallace. Directed arc consistency preprocessing. In M. Meyer, editor, *Selected papers from the ECAI-94 Workshop on Constraint Processing*, number 923 in LNCS, pages 121–137. Springer, Berlin, 1995.
- M. Wilson and A. Borning. Hierarchical constraint logic programming. *J. Log. Program.*, 16(3):277–318, 1993.
- Zhao Xing and Weixiong Zhang. MaxSolver: An efficient exact algorithm for (weighted) maximum satisfiability. *Artificial Intelligence*, 164(1-2):47–80, 2005.