

Metadata of the chapter that will be visualized in SpringerLink

Book Title	A Guided Tour of Artificial Intelligence Research	
Series Title		
Chapter Title	Valued Constraint Satisfaction Problems	
Copyright Year	2020	
Copyright HolderName	Springer Nature Switzerland AG	
Corresponding Author	Family Name	Cooper
	Particle	
	Given Name	Martin C.
	Prefix	
	Suffix	
	Role	
	Division	IRIT
	Organization	Université de Toulouse
	Address	Toulouse, France
	Email	cooper@irit.fr
Author	Family Name	de Givry
	Particle	
	Given Name	Simon
	Prefix	
	Suffix	
	Role	
	Division	MIAT
	Organization	Université de Toulouse, INRA
	Address	Castanet-Tolosan, France
	Email	Simon.de-Givry@inra.fr
Author	Family Name	Schiex
	Particle	
	Given Name	Thomas
	Prefix	
	Suffix	
	Role	
	Division	MIAT
	Organization	Université de Toulouse, INRA
	Address	Castanet-Tolosan, France
	Email	Thomas.Schiex@inra.fr

Abstract

As an extension of constraint networks, valued constraint networks (or valued CSPs) define a unifying framework for modelling optimisation problems over finite domains in which the cost domain (also denoted as the *valuation structure*) can be symbolic or numeric but is totally ordered. As is the case for constraint networks, valued CSPs can be viewed as graphical models, defined by a set of variables with an associated finite domain and by a set of cost functions, each involving a subset of the variables. Before

reading this chapter, we strongly advise the reader to finish reading chapter “Constraint Reasoning” of this Volume, dedicated to the Constraint Satisfaction Problem.

Valued Constraint Satisfaction Problems



Martin C. Cooper, Simon de Givry and Thomas Schiex

1 **Abstract** As an extension of constraint networks, valued constraint networks (or
2 valued CSPs) define a unifying framework for modelling optimisation problems over
3 finite domains in which the cost domain (also denoted as the *valuation structure*) can
4 be symbolic or numeric but is totally ordered. As is the case for constraint networks,
5 valued CSPs can be viewed as graphical models, defined by a set of variables with
6 an associated finite domain and by a set of cost functions, each involving a subset
7 of the variables. Before reading this chapter, we strongly advise the reader to finish
8 reading chapter “Constraint Reasoning” of this Volume, dedicated to the Constraint
9 Satisfaction Problem.

10 1 Introduction

11 The modelling of a combinatorial problem as a constraint network is in general quite
12 natural: it consists in identifying the decision variables of the problem together with
13 the properties that they must satisfy to form a “solution”. In a scheduling problem,
14 for example, the variables correspond to the start times of jobs and the constraints
15 concern availability or delivery dates, priorities and durations of jobs, the capacity
16 of resources, etc.

17 In certain cases, a simple list of properties to be satisfied does not provide a
18 sufficiently accurate model. This is the case, for example, if the resulting problem
does not have any solutions (the problem is over-constrained). Indeed, the properties

M. C. Cooper (✉)
IRIT, Université de Toulouse, Toulouse, France
e-mail: cooper@irit.fr

S. de Givry · T. Schiex
MIAT, Université de Toulouse, INRA, Castanet-Tolosan, France
e-mail: Simon.de-Givry@inra.fr

T. Schiex
e-mail: Thomas.Schiex@inra.fr

© Springer Nature Switzerland AG 2020
P. Marquis et al. (eds.), *A Guided Tour of Artificial Intelligence Research*,
https://doi.org/10.1007/978-3-030-06167-8_7

1

19 represented by constraints are often of different types: they can concern physical
20 impossibilities (such as a machine can only process one job at a time) or properties
21 whose satisfaction is desirable for economic reasons (such as the product must be
22 delivered on time). Modelling all these properties as hard constraints can lead to an
23 absence of a solution. On the other hand, neglecting properties which are merely
24 desirable, but not essential, leads to feasible but unsatisfactory solutions.

25 Historically, the first generalisation of the constraint satisfaction problem (CSP)
26 using cost functions goes back no doubt to Rosenfeld et al. (1976) who considered
27 “fuzzy” CSPs in which the constraints, usually defined in terms of a set of combina-
28 tions of authorised values, are replaced by “fuzzy sets” of authorised combinations,
29 each combination having a level of membership (between 0 and 1). Since fuzzy
30 sets generalise classical sets, it is thus possible to express classical constraints (with
31 membership levels being exclusively 0 or 1) but also combinations which are “pos-
32 sible” to different degrees. The aim is then to find an assignment to the variables
33 of the problem which avoids combinations of values which are unlikely or undesir-
34 able. More precisely, the aim is to maximise the minimum membership level used (a
35 max-min problem). Dedicated constraint propagation algorithms were also proposed
36 by Rosenfeld et al. (1976).

37 This extension was followed by different additional extensions (including addi-
38 tive Shapiro and Haralick 1981 or partial Freuder and Wallace 1992 constraint net-
39 works, possibilistic constraint networks Schiex 1992, lexicographic constraint net-
40 works Fargier et al. 1993, probabilistic constraint networks Fargier and Lang 1993)
41 which use different scales of costs (integers, reals, symbols) and which judge the
42 quality of a solution by combining the costs of the combinations of values by means
43 of a specific operator in each case, this giving a corresponding specific semantics
44 to “costs”: additive costs (financial or energy, for example), priorities, probabilities,
45 etc. In each case, dedicated algorithms were proposed.

46 2 Valued Constraint Networks

47 To capture the essence of the properties and algorithms which are common to all
48 these extensions, but also to better understand why some extensions are easier to
49 deal with than others, two general frameworks were proposed: semi-ring CSP or
50 SCSP Bistarelli et al. (1995, 1997) and valued CSP or VCSP Schiex et al. (1995).
51 In both cases, the constraints of the CSP, which classify the combinations of values
52 as authorised or forbidden, are replaced by cost functions (or valued constraints)
53 allowing a finer and graduated comparison of the combinations of values. Both the
54 VCSP framework and the SCSP framework are based on an abstract cost domain, in
55 which it is possible to express total authorisation or prohibition but also intermediate
56 levels. The cost domain is equipped in each case with an algebra adapted to the
57 problems which we wish to model.

58 The link between VCSP and SCSP is simple: VCSPs are SCSPs using a totally
59 ordered cost structure. Since most properties and algorithms have been designed

60 for the totally-ordered case, which is also the most frequently used in practice, we
 61 only discuss the simpler VCSP framework. In this case, the cost structure is called a
 62 “valuation structure”.

63 2.1 Valuation Structure

64 A valuation structure V is defined by a quintuplet $V = \langle E, \oplus, <, \perp, \top \rangle$ where E
 65 is the set of possible costs, \oplus is an aggregation operator for combining costs and
 66 where $<$ defines a total order on E . \perp and \top denote respectively the minimum and
 67 the maximum elements of E ($\perp < \top$). We also have that:

- 68 • \oplus is commutative ($\forall \alpha, \beta \in E, \alpha \oplus \beta = \beta \oplus \alpha$) and associative ($\forall \alpha, \beta, \gamma \in E,$
 69 $\alpha \oplus (\beta \oplus \gamma) = (\alpha \oplus \beta) \oplus \gamma$).
- 70 • \oplus is monotone ($\forall \alpha, \beta, \gamma \in E, (\alpha \preceq \beta) \Rightarrow ((\alpha \oplus \gamma) \preceq (\beta \oplus \gamma))$)
- 71 • \perp is the neutral element for \oplus ($\forall \alpha \in E, \perp \oplus \alpha = \alpha$) and \top the absorbing element
 72 ($\forall \alpha \in E, \top \oplus \alpha = \top$).

73 The associativity and commutativity of \oplus capture the fact that only the *set* of costs
 74 aggregated by \oplus has a meaning: the result of the aggregation does not depend on the
 75 order in which costs are aggregated. Monotonicity guarantees that reducing a cost
 76 within a set of costs cannot lead to an increase in the global cost. The last two axioms
 77 (concerning the neutral and absorbing elements) allow us to capture hard constraints
 78 of the constraint network using only the costs \perp (authorised combinations) and \top
 79 (forbidden combinations).

80 This algebraic structure is close to the structure of triangular co-norm, often
 81 studied in a numerical context with $E = [0, 1]$ Klement et al. (2000). It is also
 82 known as a tomonoid in algebra and includes tropical algebra Gondran and Minoux
 83 (2008).

84 Table 1 gives several examples of cost structures which are valuation structures.
 85 The following properties have important consequences concerning the algorithms
 86 for the corresponding VCSP:

- 87 1. Idempotency of \oplus ($\forall a \in E, a \oplus a = a$): we know that the only operator \oplus ful-
 88 filling this condition (together with the axioms of a valuation structure) is \max
 89 (for the order $>$) which also corresponds to \wedge (logical and) in the classical CSP.
 90 This property considerably simplifies inference because, as constraint propaga-
 91 tion does in classical CSPs, it allows us to add implicit information in a network
 92 explicitly, while preserving its meaning. On the other hand, it leads to undesirable
 93 effects in terms of modelling (the “drowning effect” of possibilistic VCSPs Schiex
 94 1992). Classical CSPs and possibilistic/fuzzy CSPs are idempotents.
- 95 2. Strict monotonicity (SM) of \oplus ($\forall a, b, c \in E, (a < c) \wedge (b \neq \top) \Rightarrow (a \oplus b) <$
 96 $(c \oplus b)$) guarantees that in the situation where the costs are not already intoler-
 97 able (equal to \top), any local cost increase leads to an increase in the global cost.
 98 This is an attractive property in terms of modelling but which renders inference

Table 1 Some classical valuation structures. SM indicates that \oplus is strictly monotonic, Idemp. that it is idempotent. The definition of \ominus is given when it exists and is simple. In the lexicographic case, $[0, 1]^*$ represents the set of multi-sets of reals between 0 and 1, \cup represents the union of multi-sets and $>_{lex}$ the lexicographic comparison of sequences obtained by sorting multi-sets in decreasing order. Thus $\{1, 0.2, 1\} >_{lex} \{0.8, 1, 0.8\}$ because in the lexicographical order $(1, 1, 0.2)$ is larger than $(1, 0.8, 0.8)$. Although the lexicographic structure is not fair, it is possible to embed it in a larger structure which is fair. See Cooper and Schiex (2004) for more details

Name	E	$a \oplus b$	$<$	\perp	\top	SM	Idemp.	$a \ominus b$
Classical	$\{t, f\}$	$a \wedge b$	$t < f$	t	f	Yes	Yes	a
Additive	$\bar{\mathbb{N}}$	$a + b$	$<$	0	$+\infty$	Yes	No	$a - b$
Weighted	$\{0, \dots, m\}$	$\min(m, a + b)$	$<$	0	m	No	No	$(a = m?m : a - b)$
Probabilistic	$[0, 1]$	$a \times b$	$>$	1	0	Yes	No	a/b
Possibilistic	$[0, 1]$	$\max(a, b)$	$<$	0	1	No	Yes	$\max(a, b)$
Fuzzy	$[0, 1]$	$\min(a, b)$	$>$	1	0	No	Yes	$\min(a, b)$
Lexico.	$[0, 1]^*$	\cup	$>_{lex}$	\emptyset	\top	Yes	No	NA

mechanisms significantly more complex. Indeed, any addition of information to a network in this case changes its meaning. Additive, probabilistic, and lexicographic VCSPs are strictly monotone.

It was to avoid the “loss of equivalence” of non-idempotent operators that a pseudo-difference operator \ominus was introduced in valuation structures. These structures are then called “fair” Schiex (2000), Cooper and Schiex (2004). We impose that for each $\beta \preceq \alpha$, there exists a maximum γ such that $\beta \oplus \gamma = \alpha$ (such a γ may not exist in infinite valuation structures). This element γ is denoted $\alpha \ominus \beta$ and defines the operator \ominus . This extra axiom is satisfied by the majority of cost structures used in practice (or can be satisfied by embedding the original valuation structure in a larger structures which is fair Cooper 2003, Cooper and Schiex 2004). The pseudo-difference operator defined in this way allows us to restore equivalence after the addition of implicit information in a network (see Sect. 5) and hence to extend the notion of constraint propagation to the non-idempotent case.

Not long afterwards, Cooper and Schiex (2004), Cooper (2005) showed that any fair and countable valuation structure can be viewed as a stack of additive/weighted valuation structures, which interact with each other as an idempotent structure (thus using $\oplus = \max$). The case $\oplus = \max$ being very close to the case of classical CSPs (Meseguer et al. 2006, p. 293), most of the work has since then focused on the weighted case.

2.2 Cost Function Networks

As for classical constraint networks, valued constraint networks (or cost function networks) define a class of graphical models.

A network \mathcal{P} is defined by a quadruplet $\langle X, D, F, V \rangle$ where X is a collection of variables, D being composed of the set of domains D_i of each of the variables $x_i \in X$. V is a valuation structure and F a set of local cost functions with values in the set E of the valuation structure V . Each cost function $f_S \in F$ (also called valued or soft constraint) is defined on the set $S \subseteq X$ of variables and specifies the cost associated with each combination of values (or *tuple*) that the variables of S can take. S is called the scope of f_S and $|S|$ is its arity (number of variables that it involves). We use $\ell(S) = \prod_{x_i \in S} D_i$ to denote the Cartesian product of the domains of the variables in S , i.e., the set of possible combinations of values for the variables of S . For a combination of values $t \in \ell(S)$ and any $T \subseteq S$, we use $t[T]$ to denote the combination of values assigned to the variables T in t (called the projection of t onto T). For simplicity of notation, unary cost functions (involving a single variable x_i) will be noted f_i and binary cost functions f_{ij} . We often also use a cost function f_\emptyset , involving no variables and representing a constant cost function.

The set of variables X and the set of scopes of the different cost functions $f_S \in F$ define a hyper-graph whose vertices are the variables of X and the hyper-edges the different scopes. In the case in which the maximum arity is limited to 2, this hyper-graph is a graph, called the constraint graph or problem graph (hence the name graphical model).

Since this hyper-graph only captures the overall structure of the problem but neither the domains nor the cost functions, a second more fine-grained representation, known as the micro-structure graph, is often used to represent (binary) VCSPs. In the micro-structure graph, the vertices correspond to the domain values of each of the variables. The vertex associated with value $a \in D_i$ is labelled by $f_i(a)$ if this cost function is defined and $f_i(a) \neq \perp$. Similarly, an edge linking vertices $a \in D_i$ and $b \in D_j$ exists and is labelled by $f_{ij}(a, b)$ if the cost function f_{ij} is defined and $f_{ij}(a, b) \neq \perp$ (see below Fig. 1).

The set of local cost functions f_S of a network implicitly defines a cost function on the set of all variables X . This global (or joint) cost function is defined simply by combining, via the aggregation operator \oplus , the set of costs produced by each of the local cost functions. Thus, the global cost function of the VCSP \mathcal{P} is defined as

$$Val_{\mathcal{P}}(t) = \bigoplus_{f_S \in F} f_S(t[S]) \text{ with } t \in \ell(X)$$

The central problem of a VCSP is to find a preferred assignment t to all its variables, i.e., such that the cost $Val_{\mathcal{P}}(t)$ is minimum (recall that the set of costs is totally ordered). This is an NP-hard problem (with the corresponding decision problem being NP-complete).

Fig. 1 The micro-structure of the problem

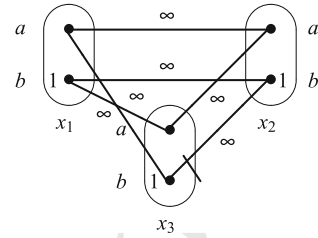


Table 2 The three cost functions coding the binary constraints

f_{12}	a	b	f_{23}	a	b	f_{13}	a	b
a	$+\infty$	0	a	$+\infty$	0	a	0	$+\infty$
b	0	$+\infty$	b	0	$+\infty$	b	$+\infty$	0

158 We say that two networks \mathcal{P} and \mathcal{P}' , defined on the same variables, are equivalent
 159 if they define the same global cost function: $Val_{\mathcal{P}} = Val_{\mathcal{P}'}$.

160 To illustrate these definitions, consider, for example, an additive CSP. In this
 161 example, the set of possible costs is the set $\bar{\mathbb{N}}$ of natural numbers together with
 162 $+\infty$, the aggregation operator \oplus is integer addition, the minimum element of this
 163 set is 0 and the maximum element $+\infty$. We consider a VCSP on three variables
 164 $X = (x_1, x_2, x_3)$. The domain of these variables is composed of two values $D_1 =$
 165 $D_2 = D_3 = \{a, b\}$. We prefer that the variables take the value a if possible and we
 166 insist that $x_1 \neq x_2$, $x_2 \neq x_3$ and $x_1 = x_3$. This problem can be modelled via six cost
 167 functions. The unary functions $f_1 = f_2 = f_3$ are identical and defined by $f_i(a) = 0$
 168 and $f_i(b) = 1$, to indicate our preference for the value a . Three functions f_{13} , f_{12}
 169 and f_{23} will capture the constraints $x_1 = x_3$, $x_1 \neq x_2$ and $x_2 \neq x_3$. Since these are
 170 hard constraints, the corresponding cost functions only use two specific costs: the
 171 cost 0 (the neutral element for $\oplus = +$, and the minimum in the scale of costs) will be
 172 associated with the authorised combinations of values and the cost $+\infty$ (the absorbing
 173 element for $\oplus = +$, and the maximum in the scale of costs) will be associated with
 174 the forbidden combinations. The tables below describe the three functions.

175 This same network is illustrated in Fig. 1 by its micro-structure. The assignment
 176 (a, a, a) has a global cost of $+\infty$ because $f_{12}((a, a)) = +\infty$. An optimal solution
 177 is (a, b, a) , of cost 1 (cost generated by f_2 since $x_2 = b$) (Table 2).

Table 3 The aggregation of f_1 and f_{12}

$f_1 \oplus f_{12}$	a	b	(x_1)
a	$+\infty$	1	
b	0	$+\infty$	
(x_2)			

2.3 Operations on the Cost Functions

A very powerful toolbox on cost function networks can be built from only simple operations. These are instantiation (or conditioning), aggregation and variable elimination.¹

Given a cost function f_S , it is possible to instantiate a variable $x_i \in S$ with one of the values a of its domain D_i . We thus obtain a cost function on $T = S - \{x_i\}$ defined by $g_T(t) = f_S(t \cup \{x_i = a\})$. In our previous example, instantiating x_1 with the value a transforms the binary function f_{12} into a unary function g_2 such that $g_2(a) = f_{12}(a, a) = +\infty$ and $g_2(b) = f_{12}(a, b) = 0$. Note that if we instantiate all the variables of a cost function, we obtain a function with an empty scope, like f_\emptyset . This operation has negligible complexity (we directly access a part of the original cost function).

The second operation is an operation involving two cost functions f_S and $g_{S'}$. This is the equivalent of the relational join operation in a CSP: it combines two cost functions into a single function, which is equivalent to the aggregation by \oplus of the two original functions. The resulting function has the scope $S \cup S'$ and is defined by $(f_S \oplus g_{S'})(t) = f_S(t[S]) \oplus g_{S'}(t[S'])$. The calculation of the aggregation of the two functions is exponential in time and space ($O(d^{|S \cup S'|})$). In our previous example, the aggregation of f_1 and f_{12} produces a function g_{12} , defined in Table 3. Observe that we can replace a set of functions by their aggregation without changing the meaning of the cost function network (i.e., without changing the global cost function Val_\emptyset).

Finally, the variable-elimination operation, consists in summarising the information from the cost function f_S on the subset of variables $T \subset S$ under the hypothesis that the eliminated variables $S - T$ are instantiated in each case in the best possible way (so as to minimise the cost).

The projection $f_S[T]$ of f_S onto T which results from the elimination of the variables of $S - T$ is the cost function $g_T(u) = \min_{v \in \ell(S-T)} f_S(u \cup v)$. Since variable-elimination requires exhausting, in general, over the whole domain of f_S , this operation has a temporal complexity $O(d^{|S|})$. The resulting function requires $O(d^{|T|})$ space. If $S - T$ is a singleton $\{x_i\}$, we will denote by $f_S[-x_i] = f_S[S - \{x_i\}]$ the result of the elimination of the single variable x_i .

If $f_S[T]$ has a cost α on an assignment of the variables T , this means that it is possible to complete this assignment on the variables $S - T$ to build an assignment

¹This latter operation is also often called projection. We have avoided this terminology because of a possible confusion with the homonymic notion introduced in Sect. 5.

211 to the variables S which has the cost α for f_S . Thus, in our running example, the
 212 elimination of x_2 in $f_{12}[-x_2]$ produces a unary function on $\{x_1\}$, uniformly equal to
 213 0. For example, it is possible to extend $x_1 = a$ (of zero cost for $f_{12}[-x_2]$) by $x_2 = b$
 214 which forms an assignment of the same (zero) cost for f_{12} .

215 It is, of course, possible in VCSPs to manipulate cost functions represented by
 216 other means than a simple table (for example, an analytic representation such as
 217 $f(x) = x^2$ or a compact data structure such as an automaton or a decision di-
 218 gram Darwiche and Marquis 2004, Fargier and Marquis 2007).

219 2.4 Links with Other Approaches

220 Therefore, valued CSPs define a generic framework for expressing and combin-
 221 ing preferences (whether numerical or symbolic, as discussed in chapter “Compact
 222 Representation of Preferences” of Volume 1) together with required properties (or
 223 constraints), expressed locally on a few variables, via cost functions. Strong links
 224 exist with numerous formalisms born out of artificial intelligence and operations
 225 research.

226 Notably, since the problem SAT defines a specialisation of CSP to propositional
 227 logic, VCSPs allow us to capture variants of SAT such as MaxSAT (maximise the
 228 number of clauses satisfied by an interpretation) and its weighted version or par-
 229 tial version (the non weighted clauses being considered as hard constraints). These
 230 problems are themselves close to quadratic pseudo-Boolean optimisation problems
 231 in which the aim is to find the minimum of a second-degree polynomial over 0/1
 232 variables Boros and Hammer (2002).

233 As a graphical model, VCSPs also allow us to capture Bayesian networks (see
 234 chapter “Belief Graphical Models for Uncertainty Representation and Reasoning”
 235 of this Volume: additive cost functions can be obtained by representing conditional
 236 probabilities by the negative of their logarithm), Markov Random fields Chellappa
 237 and Jain (1993) and factor graphs Kschischang et al. (2001). VCSP tools have demon-
 238 strated their efficiency for solving the maximum probability explanation (MPE) prob-
 239 lem in these domains in international competitions. However, in these formalisms,
 240 problems are generally not restricted to optimisation but also concern counting (or
 241 discrete integration) for which little work exists on VCSPs.

242 Looking wider afield, other formalisms for expressing preferences and utilities
 243 such as GAI (*Generalized Additive Independence* models Bacchus and Grove 1995)
 244 can be assimilated to VCSPs and hence sometimes use the techniques and solving
 245 tools developed in the VCSP framework. Connections with more distant frameworks
 246 for expressing preferences, such as “*Ceteris Paribus*” networks (or CP-nets) have also
 247 been exhibited Domshlak et al. (2003).

3 Dynamic Programming and Variable Elimination

A first approach for solving VCSPs consists in using non-serial dynamic programming Bertelé and Brioshi (1972), also called variable-elimination algorithms and better known in the constraints community under the names of *Bucket Elimination* Dechter (1999) and *Cluster Tree Elimination* Dechter (2003). These techniques, which were initially adapted to the CSP framework, extend very naturally to VCSPs. We will only present here the simplest variable-elimination technique which proceeds variable by variable, known as “*Bucket Elimination*”. Dynamic programming techniques extend naturally to a variety of problems beyond the basic optimisation problem (such as counting problems, for example). See for example Shafer and Shenoy (1988), Aji and McEliece (2000) and Dubois and Prade (1991) for the possibilistic case.

If we have a VCSP defined by a set of variables X and a set of cost functions F , the variable-elimination algorithm consists in reducing iteratively the number of variables while conserving the cost of an optimal solution. Finally, we end up with a problem having 0 variables and a cost function f_{\emptyset} equal to the cost of an optimal solution of the original problem.

Without loss of generality, suppose that the algorithm eliminates the variables in the order x_1, \dots, x_n . The elimination of a variable x_i can be described as a three-stage process:

1. We define a set K_i of all the cost functions which link variable x_i to variables which come after it in the elimination order.
2. We aggregate all these cost functions into a single cost function using \oplus . The result is a cost function g_S involving x_i and all the variables linked to x_i and which come after it in the elimination order. This aggregation is equivalent to the set of cost functions K_i .
3. We then eliminate the variable x_i from g_S and we add the cost function obtained to the network. This new cost function resumes the effect of all the cost functions K_i on these variables. We thus eliminate from the network the cost functions in K_i and the variable x_i which is no longer involved in any cost function: the variable has been eliminated.

The last two steps are in general performed simultaneously which reduces a little the spatial complexity of an elimination. The problem obtained after the elimination of x_i has a new cost function but one variable less and all the cost functions K_i have been deleted. It nevertheless has the same optimum cost as the original problem due to the invariance of the optimum cost under the operators of aggregation and elimination. In some particular cases, known as *context-specific conditional independences* in probabilistic graphical models, the new cost function can be decomposed into a set of smaller arity cost functions, reducing memory space and improving further variable eliminations Favier et al. (2011).

The order in which the variables are eliminated can have a strong influence on the efficiency of the whole process. It is possible to estimate the global complexity of the

290 procedure. Indeed, each elimination creates a new cost function but whose scope is
 291 known. As each step is exponential in the number of variables in the neighbourhood
 292 of the eliminated variable x_i and which come after x_i in the elimination order, we
 293 can simulate the process and estimate the global complexity without actually per-
 294 forming the calculations. It is the step for which the number w of neighbours (of
 295 the eliminated variable x_i which come after x_i in the elimination order) is maximum
 296 which determines the complexity (spatial and temporal): this complexity is exponen-
 297 tial in w . The parameter w is called the induced width or treewidth of the constraint
 298 graph for the given elimination order. Minimising w is an NP-hard problem but good
 299 heuristics exist Bodlaender and Koster (2008).

300 These elimination procedures are used in many domains and allow us to deal
 301 with optimisation but also counting problems, for example. Because of their spatial
 302 complexity, they are, in general, employed only in problems for which a small w
 303 exists.

304 **3.1 Partial Variable Elimination or “Mini-Buckets”**

305 Variable-elimination techniques are often inapplicable due to their high spatial (and
 306 temporal) complexity when eliminating a variable with a large number of (non-
 307 eliminated) neighbours. In order to keep this complexity under control, while sacri-
 308 ficing the optimality of the procedure, we can use the following simple approach: if
 309 the set of cost functions K_i involves too large a number of variables other than x_i ,
 310 we can simply partition K_i into a set of disjoint subsets $K_{ij} = \cup K_{ij}$ each of which
 311 involves a number of variables bounded by a given integer z .

312 We can then process each of the sets K_{ij} separately (applying the aggregation and
 313 elimination steps on each set K_{ij}). Since these sets involve a bounded number of
 314 variables, the operations have a bounded spatial and temporal complexity (which is
 315 exponential in z). Of course, the good properties of variable elimination are lost since
 316 it is not necessarily the same value of x_i which optimises each of the K_{ij} : the final cost
 317 obtained after a series of “mini-eliminations” is only a lower bound on the optimum
 318 cost. The intermediate cost functions generated during the processing of the K_{ij} thus
 319 allow us to calculate a lower bound on the cost of an optimum solution which can
 320 then be exploited in tree search algorithms using a static variable ordering Dechter
 321 and Rish (2003) or a dynamic one Marinescu and Dechter (2005, 2007). A major
 322 strength of this approach is that the lower bounds are adjustable via the parameter z .
 323 It is therefore possible to adapt it to the problem under consideration. However, this
 324 is also a disadvantage in practice because there are no theoretical or empirical results
 325 to help us to choose a priori a good value z (to obtain a good compromise between
 326 time and space).

327 Another simple and effective approach to keep the complexity of variable elimina-
 328 tion under control is to eliminate only variables having a small number of neighbour-
 329 ing variables at each node of a tree search algorithm (using an arbitrary elimination
 330 order at each node is optimal in the number of eliminations for $z \leq 2$) Larrosa (2000).

4 Search for Optimal Solutions

Given a VCSP, the central problem which is usually considered consists in identifying an assignment to the variables of minimum cost. One of the most popular approaches consists in using, as for CSPs, a depth-first tree search algorithm. The “Backtrack” algorithm for CSPs is replaced by a *Branch and Bound* algorithm. This algorithm supposes that it is possible, by a method which needs to be specified, to calculate a lower bound $lb_{\mathcal{P}}$ on the cost of an optimal solution to any VCSP. From the initial VCSP, the algorithm consists in comparing the lower bound $lb_{\mathcal{P}}$ and the cost of the best solution found so far. If $lb_{\mathcal{P}}$ is equal to or greater than this cost, then there is no point continuing the search in this branch since it is impossible to do better than the best solution found so far.

Otherwise, we have to simplify the problem. Usually, we choose one variable x_i of the problem and we exploit the fact that it must take one of the values from its domain. We thus instantiate x_i with a value chosen from its domain (conditioning). This reduces the scopes of certain cost functions and decreases the number of variables in the problem. We continue recursively by choosing a new variable. If the lower bound is greater than or equal to the cost of the best solution found so far, we reconsider the last choice we made. If it is possible to instantiate all the variables with a cost less than that of the best known solution, we have a new best solution and the search continues after updating the cost of the best solution seen so far.

When we reconsider a choice, several strategies are possible. We can simply try a new value for the most recently instantiated variable x_i ; this known as k -ary branching. We can also simply impose that the variable x_i does not take the last value tried (by deleting it from the domain of x_i). This is called binary branching, and is theoretically more powerful Mitchell (2003). Other strategies are possible.

Whatever the strategy, the choice of the next variable to instantiate or its next value can have a strong influence on the efficiency of the algorithm and different heuristics have been proposed. Concerning the choice of variable, since all variables must be assigned, we will choose the one which has the most chance to rapidly lead to the detection of the impossibility of improving the cost (e.g., the variable involved in the largest number of cost functions). The choice is of considerable importance and can capture a form of intelligent backtracking Lecoutre et al. (2006). In the CSP framework, the choice of value is difficult and is generally considered to be dependant on the application domain. In VCSPs, the costs given by the unary cost function on the current variable can guide search and we will thus choose a value minimising this cost.

The practical efficiency of this exact algorithm depends on the quality of the lower bound $lb_{\mathcal{P}}$. A simple lower bound can be obtained by aggregating the set of cost functions with empty scope created by each instantiation. This defines a constant cost which is a lower bound. But this lower bound (corresponding to the backtracking algorithm for CSPs) is too naive. In practice, the lower bound must be easy to calculate but as close as possible to the optimum, two contradictory aims since the VCSP optimisation problem is NP-hard.

Historically, various *ad hoc* procedures for calculating the lower bound were proposed. But in the last ten or fifteen years, we can observe a convergence with classical CSP approaches: “constraint propagation” procedures (also known as local consistency filtering) generalised to VCSPs which are used to calculate a lower bound. These methods work by transforming a given VCSP \mathcal{P} into another VCSP \mathcal{P}' on the same variables and which is equivalent ($Val_{\mathcal{P}} = Val_{\mathcal{P}'}$) but with an increased cost function f_{\emptyset} . This constant cost defines the lower bound, thanks to the monotonicity of \oplus . This approach by transformation has the advantage of being incremental: the transformed version can again be used by other processes. It is the hybridisation of search algorithms and valued-constraint propagation algorithms (to calculate the lower bound) which, in general, gives the best results in practice over various types of VCSP instances. In certain cases, the use of lower bounds produced by the “mini-buckets” Dechter (1997) or other algorithms Verfaillie et al. (1996) can provide an interesting alternative.

Other tree search procedures can be defined: instead of traversing the search tree in a depth-first manner, we can give the priority to the exploration of the best current partial solution (best-first search), an approach which has good properties in terms of the search tree size but can have high space requirements, possibly overcome by using a hybrid depth-first/best-first approach Allouche et al. (2015). More complex schemes have also been considered, inspired by dynamic programming Larrosa (2000), Verfaillie et al. (1996), Terrioux and Jegou (2003), Marinescu and Dechter (2005), de Givry et al. (2006), Sanchez et al. (2009), Allouche et al. (2010), or others designed to give rapidly a good quality solution (limited discrepancy search Harvey and Ginsberg 1995 and partial search de Givry and Jeannin 2006).

Local search methods and meta-heuristics Aarts and Lenstra (1997) provide an alternative which is easy to implement in order to solve VCSPs, but without any guarantee of optimality, and with the difficulty that hard constraints can be generated (forming a barrier in the landscape that may destroy ergodicity in search). These optimisation methods are very generic and few developments specific to VCSPs have been proposed, except INCOP Neveu et al. (2004), and in the domain of large/variable neighbourhood methods Loudni and Boizumault (2006), Fontaine et al. (2013), Ouali et al. (2015, 2017) for which complete search methods for VCSPs can be used to explore a large neighbourhood.

5 Propagation of Valued Constraints

In a classical CSP, we have to find an assignment to n variables which satisfies a set of constraints, where each constraint specifies the combinations of values that we can assign to a subset of the variables. The notion of constraint propagation can reduce the size of domains, tighten constraints or prune the search tree through the detection of local inconsistencies between the constraints and the domains. This notion is omnipresent in solution algorithms for CSPs. For example, during the search for a 3-colouring of a graph, as soon as we detect the existence of a vertex with three

neighbouring vertices assigned three different colours, we can prune this branch of the search tree.

Classical constraint propagation operations in CSPs translate into the propagation of implicit prohibitions corresponding to a cost \top in valued CSPs. The set of local consistency notions in VCSPs thus naturally include notions inherited from the CSP framework, but considerably enriched through notions involving a lower or upper bound.

In VCSPs known as weighted CSPs in which the aggregation operation \oplus is $+_m$, the addition operation with a ceiling $m = \top$

$$\forall \alpha, \beta \in \{0, 1, \dots, m\} \quad \alpha +_m \beta = \min\{\alpha + \beta, m\}$$

new hard constraints can appear during cost propagation. For example, if $f_i(a) +_m f_\emptyset = m$, then increasing $f_i(a)$ to m (which amounts to eliminating a from the domain D_i since $m = \top$) leaves the function Val_\emptyset invariant. In a similar way, if $f_{ij}(a, b) +_m f_i(a) +_m f_j(b) +_m f_\emptyset = m$, then we can assign the maximal cost m to $f_{ij}(a, b)$. The resulting VCSP defines exactly the same global cost function since the valuation \top is idempotent ($\top \oplus \top = \top$).

The propagation of a non-idempotent cost α must be compensated for the function Val_\emptyset to remain invariant. For example, if the aggregation operator is integer addition, then we can increase the lower bound f_\emptyset by $\alpha = \min\{f_i(a) : a \in D_i\}$ provided we decrease $f_i(a)$ by α for each value $a \in D_i$. This operation is called *unary projection*. A VCSP instance is *node consistent* when it is no longer possible to apply unary projections nor propagate an idempotent cost from f_\emptyset to a unary cost function.

We call *projection* any shifting of costs from one cost function of scope S ($|S| > 1$) to a unary cost function $f_i(a)$ (where $i \in S$ and $a \in D_i$). The increasing of $f_i(a)$ by α is compensated by the decreasing of each $f_S(t)$ (such that $t[x_i] = a$) by α . The new cost must belong to the valuation structure. For example, in the case of real costs, a necessary condition for us to be able to use f_\emptyset as a lower bound of Val_\emptyset is to have non-negative costs, which guarantees the monotonicity of \oplus . It follows that the projected cost α must not be greater than $\min\{f_S(t) : t[x_i] = a\}$. A VCSP instance is *arc consistent* if it is node consistent and if it is not possible to apply a projection nor to propagate an idempotent cost to a constraint f_S . In general, the arc consistency closure is not unique. Thus, several different arc consistency notions co-exist in the VCSP framework according to the time we are prepared to devote to finding the best closure.

We call *extension* any shifting of a cost α from $f_i(a)$ to f_S (where $|S| > 1$). In the case of real costs, an extension is equivalent to the projection of a negative cost. A well-chosen set of projection and extension operations can group together costs in a single variable, which can trigger an increase in the lower bound f_\emptyset via a unary projection. Figure 2a shows the microstructure graph of a VCSP consisting of two variables with domains $\{a, b\}$. There are two unary cost functions ($f_1(a) = f_2(a) = 1$), and the edge which links the two values b represents a cost of 1: $f_{1,2}(b, b) = 1$. Figure 2b shows the result of an extension from $f_2(a)$ to the binary cost function, whereas Fig. 2c shows the result of a projection from this binary cost function to

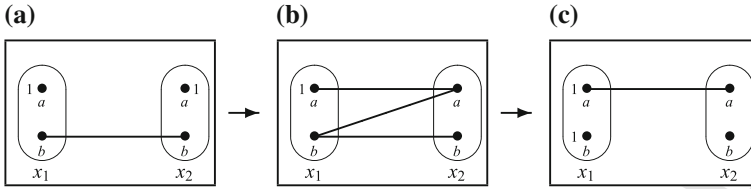


Fig. 2 Extension and projection operations applied to a VCSP

458 $f_1(b)$. Finally, a unary projection from the unary cost function f_1 would allow us to
 459 deduce a lower bound of $f_\emptyset = 1$. When costs belong to $\mathbb{Q}^{\geq 0} \cup \{\infty\}$, it is possible to
 460 find a set of operations producing the best possible increase of f_\emptyset by solving a linear
 461 program Schlesinger (1976), Cooper et al. (2007, 2010). This operation is known
 462 as *optimal soft arc consistency* (OSAC). Unfortunately, for the moment, the size of
 463 the linear program that needs to be solved limits this technique to the preprocessing
 464 step.

465 We can, however, get close to the optimal closure without having to solve a linear
 466 program. Given a VCSP instance \mathcal{P} , we can transform it into a CSP instance by
 467 replacing each cost function f_S (except f_\emptyset) by the relation

$$468 \quad R_P = \{t \in \ell(S) : f_S(t) = 0\}$$

469 We call the resulting CSP instance $\text{Bool}(\mathcal{P})$ because its valuation structure has
 470 become Boolean. The solutions to $\text{Bool}(\mathcal{P})$ are assignments t such that $\text{Val}_{\mathcal{P}}(t) =$
 471 f_\emptyset . The VCSP instance \mathcal{P} is *virtual arc consistent* (VAC) if the closure by arc
 472 consistency (in the CSP sense) of $\text{Bool}(\mathcal{P})$ is non-empty. Let S be a sequence of
 473 constraint propagation operations (in the CSP sense) which eliminates all the values
 474 in some domain D_i of $\text{Bool}(\mathcal{P})$. It is always possible to translate S into a sequence S'
 475 of projections, extensions and unary projections which increase f_\emptyset in \mathcal{P} Cooper et al.
 476 (2008, 2010). To establish virtual arc consistency, it suffices to look for a sequence S ,
 477 to apply the corresponding S' and to start over again until the arc-consistency closure
 478 of $\text{Bool}(\mathcal{P})$ is non-empty. Even if, in theory, convergence is not guaranteed in finite
 479 time, in practice, this algorithm is sufficiently fast to be applied during search. Since
 480 it is quite complex, it is no doubt open to many improvements.

481 A technique which is heuristic but less time-consuming tries to group together
 482 costs by always sending them towards variables which appear earlier in the instanti-
 483 ation order. We can guarantee convergence by allowing only those extension opera-
 484 tions which respect this order. EDAC combines this directional approach with a local
 485 version of VAC on the neighbourhood of each variable de Givry et al. (2005).

486 Up until now, most research has been concentrated on forms of arc consistency,
 487 based only on cost movements involving a single cost function of arity 2
 488 or more Sánchez et al. (2008). There remains theoretical, algorithmic and experi-
 489 mental research to be done to explore the world of higher-level consistency notions,
 490 involving several cost functions of arity 2 or more Nguyen et al. (2017).

491 *Global Cost Functions*

492 Part of the success of constraint programming arise from the introduction of so-called
 493 “global constraints”. Global constraints are constraints with a restricted semantics
 494 that can be exploited in order to define constraint propagation algorithms which are
 495 more efficient than the generic algorithms (exponential time in the arity). This can
 496 also be achieved with some cost functions if their minimum can be identified more
 497 efficiently than in the general case: efficient bound consistency algorithms have been
 498 proposed in these cases Zytnicki et al. (2009).

499 A famous example of a global constraint is the “AllDiff” constraint which
 500 requires a set of variables to take all different values (as introduced in chapter “Con-
 501 straint Reasoning” of this Volume). This constraint’s semantics admits an efficient
 502 propagation algorithm based on matching theory.

503 If various soft “global constraints” have been proposed that rely on the introduction
 504 of extra cost variables, it is only quite recently that it has been shown how cost
 505 propagation on global cost functions similar to AllDiff can be achieved Lee and
 506 Leung (2009, 2012). This result paves the way for a long series of results targeting
 507 the transfer of the list of existing hard global constraints to VCSP. Specifically, this
 508 was achieved for global cost functions which are decomposable in well-organised
 509 networks of cost functions of bounded arity Allouche et al. (2012, 2016).

510 Another approach is to transfer cut generation techniques from integer program-
 511 ming to VCSP, like in the clique global constraint de Givry and Katsirelos (2017).

512 **6 Complexity and Tractable Classes**

513 The complexity of the VCSP depends on its valuation structure. However, every
 514 valuation structure contains an idempotent valuation \perp (which represents total satis-
 515 faction and hence a zero cost) and an absorbing valuation $\top \neq \perp$ (corresponding to
 516 a totally unsatisfactory and hence forbidden assignment). Since it is always possible
 517 to code all CSP instances via the valuations \perp and \top , the VCSP is NP-hard for any
 518 valuation structure.

519 Nevertheless, there are polynomial-time algorithms for certain subproblems of the
 520 VCSP and for certain valuation structures. If the aggregation operator is idempotent
 521 (which is the case in fuzzy and possibilistic CSPs), the study of the complexity of
 522 the VCSP reduces to the study of the complexity of the CSP, because solving the
 523 VCSP comes down to solving cut problems (Meseguer et al. 2006, p. 293), which
 524 we obtain by replacing each cost function f_P by the relation

$$525 \quad R_P^\alpha = \{t \in I(P) : f_P(t) \leq \alpha\}.$$

526 when the aggregation operator is not idempotent, the study of the complexity of
 527 the VCSP does not follow directly from the study of the complexity of the CSP.
 528 For example, 2SAT $\in P$ but MAX-2SAT is NP-hard. There is only one non-trivial
 529 class of cost functions defining a tractable subproblem of the VCSP which comes

530 out of the study of MAX-SAT: the class of submodular functions. Let $f_P : D' \rightarrow$
 531 $\mathbb{R}^{\geq 0} \cup \{\infty\}$ be a cost function where the domain D has a total order. The function f_P
 532 is *submodular* Fujishige (2005) if $\forall x = (x_1, \dots, x_r) \in D', y = (y_1, \dots, y_r) \in D',$

$$533 \quad f_P(\min(x, y)) + f_P(\max(x, y)) \leq f_P(x) + f_P(y)$$

534 where $\min(x, y) = (\min(x_1, y_1), \dots, \min(x_r, y_r))$ and $\max(x, y)$ is defined simi-
 535 larly. The class of submodular functions includes all unary functions, the binary
 536 functions $\sqrt{x^2 + y^2}$, $((x \geq y) ? (x - y)^t : m)$ (for $t \geq 1$), $K - xy$, the cut function
 537 of a graph, the rank function of a matroid (Gondran and Minoux 2009, Chap. 9) as
 538 well as the function η_a^ρ (for $\rho > 0$), where

$$539 \quad \eta_a^\rho(x) = \begin{cases} 0 & \text{if } (x_1 < a_1 \wedge \dots \wedge x_r < a_r) \vee (x_{r+1} > a_{r+1} \wedge \dots \wedge x_s > a_s) \\ \rho & \text{otherwise.} \end{cases}$$

540 An algorithm developed in Operations Research solves VCSPs with submodular cost
 541 functions in $O(n^3 e \log^2 n + n^4 \log^{O(1)} n)$ time, where n is the number of variables
 542 and e the number of cost functions Lee et al. (2015). Another approach Cooper
 543 et al. (2008) consists in establishing virtual arc consistency (which preserves the
 544 submodularity of the cost functions). By the definition of VAC, the arc-consistency
 545 closure \mathcal{Q} of $\text{Bool}(\mathcal{P})$ is non-empty and, by definition of submodularity, its cost
 546 functions are both *min-closed* and *max-closed* Jeavons and Cooper (1995); to find a
 547 solution of \mathcal{Q} (which is necessarily an optimal solution of \mathcal{P}), it suffices to assign
 548 always the minimum (or always the maximum) value in each domain of \mathcal{Q} .

549 A VCSP can be coded as an integer programming problem (whose variables
 550 include $v_{ia} \in \{0, 1\}$ which is equal to 1 if and only if $x_i = a$ in the original VCSP
 551 instance Hurley et al. 2016). The linear relaxation of this integer programming prob-
 552 lem (in which v_{ia} is now a real number in the interval $[0, 1]$) has integer solutions
 553 if all cost functions are submodular, meaning that the VCSP can be solved by lin-
 554 ear programming. The dual of this relaxation is exactly the linear program used by
 555 OSAC Cooper et al. (2010) to transform the original instance into an equivalent
 556 instance with an explicit lower bound on costs Werner (2007); for instances with
 557 submodular cost functions, this explicit lower bound is thus equal to the cost of
 558 an optimal solution. Indeed, the VCSP restricted to a language of finite-valued cost
 559 functions over the valuation structure $\mathbb{Q}^{\geq 0} \cup \{\infty\}$ is tractable if and only if it is solved
 560 by this linear program Thapper and Zivny (2013). This notably includes languages
 561 of cost functions that are submodular on arbitrary lattices. State-of-the-art results
 562 concerning the tractability of languages of cost functions are covered in detail in a
 563 recent comprehensive survey article Jeavons et al. (2014). These theoretical results
 564 demonstrate the importance of submodularity and linear programming in the search
 565 for tractable subproblems of the VCSP.

566

7 Solvers and Applications

567

Various solvers have been developed to solve either cost function networks or their specialisations such as Weighted Partial Maximum Satisfiability problems (MaxSAT). MaxSAT problems can be described as cost function networks using Boolean domains and cost functions defined using (weighted) clauses.

571

The MaxSAT international competition attracts the most famous solvers. One can easily access the associated web site at <http://maxsat.ia.udl.cat/>. These solvers are usually able to tackle problems that are formalised using weighted propositional logic using the conjunctive normal form as a set of clauses. Clauses can be weighted or hard. However, some solvers can still be restricted to non-weighted clauses or unable to express hard clauses.

577

If we consider the more general VCSP case, most existing solvers focus on weighted VCSPs, using additive costs (bounded or not) because they have a large application area and can also be used to represent stochastic graphical models such as Bayesian nets and Markov random fields. MaxCSP and graphical model solving competitions are again very useful to access all these tools.²

582

A large collection of real, academic and random cost function networks is maintained and accessible at <http://costfunction.org>. Part of this collection has been completed with MaxCSP, MaxSAT, Constraint Programming, and Markov random fields used in image processing and other problems to define a challenging collection of benchmarks for solver evaluation³ Hurley et al. (2016).

587

Several of the techniques introduced in this chapter are also implemented in the open source award-winning `toulbar2`⁴ solver which is able to process cost function networks but also problems represented as stochastic graphical models. Mini-buckets techniques combined with branch and bound form the core of the `daoopt` solver.⁵

591

Applications

592

The application target of these models and associated solvers is wide and covers any problem that can be reduced to the optimisation of a sum of local cost functions over discrete variables. They have been used among others to tackle resource management problems (such as frequency assignment problems Cabon et al. 1999 or observation satellite scheduling Verfaillie et al. 1996), cost-optimal planning Cooper et al. (2006), structured RNA gene finding Zytnicki et al. (2008), complex pedigree analysis Sánchez et al. (2008), haplotype reconstruction Favier et al. (2010), genetic analysis Silberstein et al. (2013), protein design Allouche et al. (2014); Traoré et al. (2013); Simoncini et al. (2015), crop allocation problems Akplogan et al.

²See <http://www.cril.univ-artois.fr/CPAI08/>, <http://www.cs.huji.ac.il/project/UAI10/>, <http://www.cs.huji.ac.il/project/PASCAL/index.php>, <http://www.hlt.utdallas.edu/~vgogate/uai14-competition/>, <http://www.hlt.utdallas.edu/~vgogate/uai16-evaluation/>.

³See <http://genoweb.toulouse.inra.fr/~degivry/evalgm/> in LP, MINIZINC, UAI, WCNF, and WCSP formats.

⁴See <http://www.inra.fr/mia/T/toulbar2>.

⁵See <https://github.com/lotten/daoopt>.

601 (2013), judge assignment problems de Givry et al. (2014), in inductive logic pro-
 602 gramming Alphonse and Rouveiro (2007), natural language processing,⁶ flowchart
 603 structure recognition Bresler et al. (2013), multi-agent planning Kumar and Zilber-
 604 stein (2010), partially observable Markov decision processes processing Dibangoye
 605 et al. (2013), model abstraction Struss et al. (2011), diagnosis Maier et al. (2011),
 606 music processing and Markov logic Papadopoulos and Tzanetakis (2012, 2013),
 607 probabilistic counting Ermon et al. (2013) and inference Ghosh et al. (2015),

608 Beyond pure discrete optimisation, Valued CSPs also capture the Maximum Prob-
 609 ability Explanation (MPE) in Bayesian networks and the maximum a posteriori
 610 (MAP) problem in Markov random fields, commonly used in image analysis Kappes
 611 et al. (2015), that are immediately reducible to additive VCSPs Hurley et al. (2016).
 612 They are therefore able to capture problems on related probabilistic models such
 613 as complex hidden Markov models or conditional random fields, as long as a finite
 614 horizon is used.

615 Beyond probabilistic criteria, less purely numerical criteria such as the “lexico-
 616 graphic” ordering that can be used over possibilistic VCSPs can also be reduced to
 617 additive VCSPs Schiex et al. (1995).

618 8 Conclusion

619 This rapid presentation obviously does not claim to be exhaustive. We have not, for
 620 example, mentioned *knowledge compilation* techniques, well known in the CSP
 621 propositional logic communities (see the chapter “Reasoning with Propositional
 622 Logic: from SAT Solvers to Knowledge Compilation” of this Volume) and which have
 623 also been extended towards optimisation Bergman et al. (2016) including
 624 VCSPs Darwiche and Marquis (2004), Fargier and Marquis (2007) or semi-ring
 625 CSPs Wilson (2005).

626 Nevertheless, this chapter shows that the VCSP formalism, within the last fifteen
 627 years, has been enriched by a set of theoretical results (tractable classes), techniques,
 628 algorithms (tree search, local consistencies, variable elimination ...) as well as soft-
 629 ware tools which make it a solid model for expressing and solving combinatorial
 630 optimisation problems on graphical models (whether deterministic or stochastic).
 631 The recent extensions of the notion of global constraint (or global cost function)
 632 based on dedicated local consistencies should provide the modeller with a tool which
 633 is as rich as in classical constraints, but with a greater expressive power and efficiency
 634 for optimisation problems. Other domains, such as the exploitation of symmetries in
 635 VCSPs and pruning by dominance de Givry et al. (2013), for example, have yet to
 636 be studied.

637 The notion of valued graphical model has, in fact, been used beyond the con-
 638 straint programming community. The notions of discrete stochastic graphical mod-
 639 els such as Bayesian networks or Markov random fields Chellappa and Jain (1993)

⁶See <https://code.google.com/p/hltidi-l3>.

640 or factor graphs Kschischang et al. (2001) are formalisms which are essentially
 641 equivalent to cost function networks but for which the probabilistic interpretation
 642 naturally begs the question of how to count the solutions—linked to the calculation
 643 of “marginals”—rather than optimisation problems. The notions of local consistency
 644 in VCSPs, preserving joint distributions, could find a new domain of application, as
 645 global cost functions could increase the expressive power of these formalisms, which
 646 are often limited, in the discrete case, to express cost functions (conditional prob-
 647 abilities, potentials, factors) involving a small number of variables and defined by
 648 tables.

649 Cost function networks thus provide an ideal tool for expressing and process-
 650 ing simultaneously preferences/utilities, uncertainties/probabilities (see the chapter
 651 “Representations of Uncertainty in Artificial Intelligence: Probability and Possibil-
 652 ity” of Volume 1) and classic feasibilities and constraints. In this framework, we
 653 usually distinguish control variables (controlled by the decider) from non-controlled
 654 variables (representing the environment and the source of uncertainty). We could
 655 refer to Dubois et al. (1996) for the possibilistic case, the algebraic treatment of such
 656 problems having been studied especially in Pralet et al. (2007).

657 References

- 658 Aarts E, Lenstra J (1997) Local search in combinatorial optimization. Interscience series in discrete
 659 mathematics and optimization, Wiley, New York
- 660 Aji S, McEliece R (2000) The generalized distributive law. *IEEE Trans Inf Theory* 46(2):325–343
- 661 Akplogan M, de Givry S, Métivier JP, Quesnel G, Joannon A, Garcia F (2013) Solving the crop
 662 allocation problem using hard and soft constraints. *RAIRO - Oper Res* 47:151–172
- 663 Allouche D, de Givry S, Schiex T (2010) Towards parallel non serial dynamic programming for
 664 solving hard weighted CSP. In: *Proceedings of CP-10, St Andrews, Scotland*, pp 53–60
- 665 Allouche D, Bessiere C, Boizumault P, de Givry S, Métivier J, Gutierrez P, Loudni S, Schiex T (2012)
 666 Filtering decomposable global cost functions. In: *Proceedings of AAAI-12, Toronto, Canada*, pp
 667 407–413
- 668 Allouche D, André I, Barbe S, Davies J, de Givry S, Katsirelos G, O’Sullivan B, Prestwich S,
 669 Schiex T, Traoré S (2014) Computational protein design as an optimization problem. *Artif Intell*
 670 212:59–79
- 671 Allouche D, de Givry S, Katsirelos G, Schiex T, Zytnicki M (2015) Anytime hybrid best-first search
 672 with tree decomposition for weighted CSP. In: *Proceedings of CP-15, Cork, Ireland*, pp 12–28
- 673 Allouche D, Bessiere C, Boizumault P, de Givry S, Gutierrez P, Lee J, Leung K, Loudni S, Métivier
 674 J, Schiex T, Wu Y (2016) Tractability-preserving transformations of global cost functions. *Artif*
 675 *Intell J* 238:166–189
- 676 Alphonse E, Rouveiroi C (2007) Extension of the top-down data-driven strategy to ILP. In: *Inductive*
 677 *logic programming*. Springer, pp 49–63
- 678 Bacchus F, Grove A (1995) Graphical models for preference and utility. In: *Proceedings of UAI-95,*
 679 *Montreal, Quebec, Canada*, pp 3–10
- 680 Bergman D, Cire AA, van Hoeve WJ, Hooker J (2016) *Decision diagrams for optimization*. Springer,
 681 Berlin
- 682 Bertelé U, Brioshi F (1972) *Nonserial dynamic programming*. Academic Press, London
- 683 Bistarelli S, Montanari U, Rossi F (1995) Constraint solving over semirings. In: *Proceedings of*
 684 *IJCAI-95, Montréal, Canada*

- 685 Bistarelli S, Montanari U, Rossi F (1997) Semiring-based constraint satisfaction and optimization.
686 J ACM 44(2):201–236
- 687 Bodlaender H, Koster A (2008) Treewidth Computations I. Upper Bounds. Technical Report UU-
688 CS-2008-032, Utrecht University, Department of Information and Computing Sciences, Utrecht,
689 The Netherlands
- 690 Boros E, Hammer P (2002) Pseudo-boolean optimization. *Discret Appl Math* 123:155–225
- 691 Bresler M, Prusa D, Hlavác V (2013) Modeling flowchart structure recognition as a max-sum
692 problem. 12th International conference on document analysis and recognition. USA, Washington,
693 DC, pp 1215–1219
- 694 Cabon B, de Givry S, Lobjois L, Schiex T, Warners J (1999) Radio link frequency assignment.
695 *Constraints* 4:79–89
- 696 Chellappa R, Jain A (1993) Markov random fields: theory and applications. Academic Press, London
- 697 Cooper M (2003) Reduction operations in fuzzy or valued constraint satisfaction. *Fuzzy Sets Syst*
698 134(3):311–342
- 699 Cooper M (2005) High-order consistency in valued constraint satisfaction. *Constraints* 10:283–305
- 700 Cooper M, Schiex T (2004) Arc consistency for soft constraints. *Artif Intell* 154:199–227
- 701 Cooper M, Cussat-Blanc S, Roquemaurel MD, Régnier P (2006) Soft arc consistency applied to
702 optimal planning. In: *Proceedings of CP-06, Nantes, France*, pp 680–684
- 703 Cooper M, de Givry S, Schiex T (2007) Optimal soft arc consistency. In: *Proceedings of IJCAI-07,*
704 *Hyderabad, India*, pp 68–73
- 705 Cooper M, de Givry S, Sanchez M, Schiex T, Zytynicki M (2008) Virtual arc consistency for valued
706 CSP. In: *Proceedings of AAAI-08, Chicago, IL*, pp 253–258
- 707 Cooper M, de Givry S, Sánchez M, Schiex T, Zytynicki M, Werner T (2010) Soft arc consistency
708 revisited. *Artif Intell* 174(7):449–478
- 709 Darwiche A, Marquis P (2004) Compiling propositional weighted bases. *Artif Intell* 157(1):81–113
- 710 de Givry S, Jeannin L (2006) A unified framework for partial and hybrid search methods in constraint
711 programming. *Comput Oper Res* 33(10):2805–2833
- 712 de Givry S, Katsirelos G (2017) Clique cuts in weighted constraint satisfaction. In: *Proceedings of*
713 *CP-17, Melbourne, Australia*, pp 97–113
- 714 de Givry S, Zytynicki M, Heras F, Larrosa J (2005) Existential arc consistency: getting closer to full
715 arc consistency in weighted CSPs. In: *Proceedings of IJCAI-05, Edinburgh, Scotland*, pp 84–89
- 716 de Givry S, Schiex T, Verfaillie G (2006) Exploiting tree decomposition and soft local consistency
717 in weighted CSP. In: *Proceedings of AAAI-06, Boston, MA*, pp 22–27
- 718 de Givry S, Prestwich S, O’Sullivan B (2013) Dead-end elimination for weighted CSP. In: *Proceed-*
719 *ings of CP-13, Uppsala, Sweden*, pp 263–272
- 720 de Givry S, Lee J, Leung K, Shum Y (2014) Solving a judge assignment problem using conjunctions
721 of global cost functions. In: *Proceedings of CP-14, Lyon, France*, pp 797–812
- 722 Dechter R (1997) Mini-buckets: a general scheme for generating approximations in automated
723 reasoning. In: *Proceedings of IJCAI-97*, pp 1297–1303
- 724 Dechter R (1999) Bucket elimination: a unifying framework for reasoning. *Artif Intell* 113(1–2):41–
725 85
- 726 Dechter R (2003) *Constraint processing*. Morgan Kaufmann Publishers, Burlington
- 727 Dechter R, Rish I (2003) Mini-buckets: a general scheme for approximating inference. *J ACM*
728 50(2):1–61
- 729 Dibangoye J, Amato C, Buffet O, Charpillet F (2013) Optimally solving Dec-POMDPs as
730 continuous-state MDPs. In: *Proceedings of IJCAI-13, Beijing, China*, pp 90–96
- 731 Domshlak C, Rossi F, Venable K, Walsh T (2003) Reasoning about soft constraints and conditional
732 preferences: complexity results and approximation techniques. In: *Proceedings of IJCAI-03,*
733 *Acapulco, Mexico*, pp 215–220
- 734 Dubois D, Prade H (1991) Inference in possibilistic hypergraphs. *Uncertainty in Knowledge Bases*
735 pp 249–259
- 736 Dubois D, Fargier H, Prade H (1996) Possibility theory in constraint satisfaction problems: handling
737 priority, preference and uncertainty. *Appl Intell* 6(4):287–309

- 738 Ermon S, Gomes C, Sabharwal A, Selman B (2013) Embed and project: discrete sampling with
 739 universal hashing. In: *Advances in Neural information processing systems*, pp 2085–2093
- 740 Fargier H, Lang J (1993) Uncertainty in constraint satisfaction problems: a probabilistic approach.
 741 In: *Proceedings of ECSQARU '93*, Grenada, Spain, vol 747, pp 97–104
- 742 Fargier H, Marquis P (2007) On valued negation normal form formulas. In: *Proceedings of IJCAI-*
 743 *07*, Hyderabad, India, pp 360–365
- 744 Fargier H, Lang J, Schiex T (1993) Selecting preferred solutions in fuzzy constraint satisfaction
 745 problems. In: *Proceedings of the 1st European congress on fuzzy and intelligent technologies*
- 746 Favier A, Elsen JM, de Givry S, Legarra A (2010) Optimal haplotype reconstruction in half-sib
 747 families. In: *ICLP-10 workshop on constraint based methods for bioinformatics*, Edinburgh, UK
- 748 Favier A, de Givry S, Legarra A, Schiex T (2011) Pairwise decomposition for combinatorial opti-
 749 mization in graphical models. In: *Proceedings of IJCAI-11*, Barcelona, Spain
- 750 Fontaine M, Loudni S, Boizumault P (2013) Exploiting tree decomposition for guiding neighbor-
 751 hoods exploration for VNS. *RAIRO - Oper Res* 47(2):91–123
- 752 Freuder E, Wallace R (1992) Partial constraint satisfaction. *Artif Intell* 58:21–70
- 753 Fujishige S (2005) Submodular functions and optimisation, vol 58, 2nd edn. *Annals of discrete*
 754 *mathematics*. Elsevier, Amsterdam
- 755 Ghosh S, Kumar A, Varakantham P (2015) Probabilistic inference based message-passing for
 756 resource constrained DCOPs. In: *Proceedings of IJCAI-15*, Buenos Aires, Argentina, pp 411–417
- 757 Gondran M, Minoux M (2008) Graphs, dioids and semirings: new models and algorithms, vol 41.
 758 Springer Science & Business Media, Berlin
- 759 Gondran M, Minoux M (2009) *Graphes et algorithmes*. Lavoisier, EDF R&D
- 760 Harvey W, Ginsberg M (1995) Limited discrepancy search. In: *Proceedings of IJCAI-95*, Montréal,
 761 Canada
- 762 Hurley B, OSullivan B, Allouche D, Katsirelos G, Schiex T, Zytnicki M, de Givry S, (2016)
 763 Multi-language evaluation of exact solvers in graphical model discrete optimization. *Constraints*
 764 21(3):413–434
- 765 Jeavons P, Cooper M (1995) Tractable constraints on ordered domains. *Artif Intell* 79(2):327–339
- 766 Jeavons P, Krokhin A, Živný S (2014) The complexity of valued constraint satisfaction. *Bull EATCS*
 767 2(113):
- 768 Kappes J, Andres B, Hamprecht F, Schnörr C, Nowozin S, Batra D, Kim S, Kausler B, Kröger T,
 769 Lellmann J, Komodakis N, Savchynskyy B, Rother C (2015) A comparative study of modern
 770 inference techniques for structured discrete energy minimization problems. *Int J Comput Vis*
 771 115(2):155–184
- 772 Klement E, Mesiari R, Pap E (2000) *Triangular norms*. Kluwer Academic Publishers, Dordrecht
- 773 Kschischang F, Frey B, Loeliger H (2001) Factor graphs and the sum-product algorithm. *IEEE*
 774 *Trans Inf Theory* 47(2):498–519
- 775 Kumar A, Zilberstein S (2010) Point-based backup for decentralized POMDPs: complexity and
 776 new algorithms. In: *Proceedings of the 9th international conference on autonomous agents and*
 777 *multiagent systems*, pp 1315–1322
- 778 Larrosa J (2000) Boosting search with variable elimination. In: *Principles and practice of constraint*
 779 *programming - CP 2000*, Singapore, pp 291–305
- 780 Lecoutre C, Sais L, Tabary S, Vidal V (2006) Last conflict based reasoning. In: *Proceedings of*
 781 *ECAI-06*, Riva del Garda, Italy, pp 133–137
- 782 Lee J, Leung K (2009) Towards efficient consistency enforcement for global constraints in weighted
 783 constraint satisfaction. In: *Proceedings of IJCAI-09*, Pasadena, USA
- 784 Lee J, Leung K (2012) Consistency techniques for flow-based projection-safe global cost functions
 785 in weighted constraint satisfaction. *J Artif Intell Res* 43:257–292
- 786 Lee Y, Sidford A, Wong S (2015) A faster cutting plane method and its implications for combinatorial
 787 and convex optimization. *IEEE 56th annual symposium on foundations of computer science*
 788 *(FOCS)*. Berkeley, CA, USA, pp 1049–1065
- 789 Loudni S, Boizumault P (2006) Combining VNS with constraint programming for solving anytime
 790 optimization problems. *Eur J Oper Res* 191(3):705–735

- 791 Maier P, Jain D, Sachenbacher M (2011) Diagnostic hypothesis enumeration vs. probabilistic inference
792 for hierarchical automata models. In: International workshop on principles of diagnosis
793 (DX), Murnau, Germany
- 794 Marinescu R, Dechter R (2005) AND/OR branch-and-bound for graphical models. In: Proceedings
795 of IJCAI-05, Edinburgh, Scotland, pp 224–229
- 796 Marinescu R, Dechter R (2007) Best-first AND/OR search for most probable explanations. In:
797 Proceedings of UAI-07, Vancouver, BC, Canada, pp 259–266
- 798 Meseguer P, Rossi F, Schiex T (2006) Soft constraints. Elsevier, Amsterdam, pp 281–328. Chap 9
799 Mitchell D (2003) Resolution and constraint satisfaction. In: Proceedings of CP-03, Kinsale, Ireland,
800 pp 555–569
- 801 Neveu B, Trombetti G, Glover F (2004) ID Walk: a candidate list strategy with a simple diversi-
802 fication device. In: Proceedings of CP-04, Toronto, Canada, pp 423–437
- 803 Nguyen H, Bessiere C, de Givry S, Schiex T (2017) Triangle-based consistencies for cost function
804 networks. *Constraints* 22:230–264. <https://doi.org/10.1007/s10601-016-9250-1>
- 805 Ouali A, Loudni S, Loukil L, Boizumault P, Lebbah Y (2015) Replicated parallel strategies for
806 decomposition guided VNS. *Electron Notes Discret Math* 47:93–100
- 807 Ouali A, Allouche D, de Givry S, Loudni S, Lebbah Y, Eckhardt F, Loukil L (2017) Iterative
808 decomposition guided variable neighborhood search for graphical model energy minimization.
809 In: Proceedings of UAI-17, Sydney, Australia, pp 550–559
- 810 Papadopoulos H, Tzanetakis G (2012) Modeling chord and key structure with Markov logic. In:
811 Proceedings international conference of the society for music information retrieval (ISMIR), pp
812 121–126
- 813 Papadopoulos H, Tzanetakis G (2013) Exploiting structural relationships in audio music signals
814 using markov logic networks. In: 38th international conference on acoustics, speech, and signal
815 processing (ICASSP), Canada, pp 4493–4497
- 816 Pralet C, Verfaillie G, Schiex T (2007) An algebraic graphical model for decision with uncertainties,
817 feasibilities, and utilities. *J Artif Intell Res* 29:421–489
- 818 Rosenfeld A, Hummel R, Zucker S (1976) Scene labeling by relaxation operations. *IEEE Trans*
819 *Syst Man Cybern* 6(6):173–184
- 820 Sánchez M, de Givry S, Schiex T (2008) Mendelian error detection in complex pedigrees using
821 weighted constraint satisfaction techniques. *Constraints* 13(1–2):130–154
- 822 Sanchez M, Allouche D, de Givry S, Schiex T (2009) Russian doll search with tree decomposition.
823 In: Proceedings of IJCAI'09, Pasadena, USA
- 824 Schiex T (1992) Possibilistic constraint satisfaction problems or “How to handle soft constraints?”.
825 In: Proceedings of UAI-92, Stanford, CA, pp 269–275
- 826 Schiex T (2000) Arc consistency for soft constraints. In: Proceedings of CP-00, Singapore, pp
827 411–424
- 828 Schiex T, Fargier H, Verfaillie G (1995) Valued constraint satisfaction problems: hard and easy
829 problems. In: Proceedings of IJCAI-95, Montréal, Canada, pp 631–637
- 830 Schlesinger M (1976) Sintaksicheskiy analiz dvumernykh zritelnykh signalov v usloviyakh pomekh
831 (Syntactic analysis of two-dimensional visual signals in noisy conditions). *Kibernetika* 4:113–
832 130. in Russian
- 833 Shafer G, Shenoy P (1988) Local computations in hyper-trees. Working paper 201, School of
834 business, University of Kansas
- 835 Shapiro L, Haralick R (1981) Structural descriptions and inexact matching. *IEEE Trans Pattern*
836 *Anal Mach Intell* 3:504–519
- 837 Silberstein M, Weissbrod O, Otten L, Tzemach A, Anisenia A, Shtark O, Tuberg D, Galfrin E,
838 Gannon I, Shalata A, Borochowitz Z, Dechter R, Thompson E, Geiger D (2013) A system for
839 exact and approximate genetic linkage analysis of SNP data in large pedigrees. *Bioinformatics*
840 29(2):197–205
- 841 Simoncini D, Allouche D, de Givry S, Delmas C, Barbe S, Schiex T (2015) Guaranteed discrete
842 energy optimization on large protein design problem s. *J Chem Theory Comput* 11(12):5980–
843 5989

- 844 Struss P, Fraracci A, Nyga D (2011) An automated model abstraction operator implemented in the
845 multiple modeling environment MOM. In: 25th international workshop on qualitative reasoning,
846 Barcelona, Spain
- 847 Terrioux C, Jegou P (2003) Bounded backtracking for the valued constraint satisfaction problems.
848 In: Proceedings of CP-03, Kinsale, Ireland, pp 709–723
- 849 Thapper J, Zivny S (2013) The complexity of finite-valued CSPs. In: Proceedings of the forty-fifth
850 annual ACM symposium on Theory of computing. ACM, pp 695–704
- 851 Traoré S, Allouche D, André I, de Givry S, Katsirelos G, Schiex T, Barbe S (2013) A new framework
852 for computational protein design through cost function network optimization. *Bioinformatics*
853 29(17):2129–2136
- 854 Verfaillie G, Lemaître M, Schiex T (1996) Russian doll search. In: Proceedings of AAAI'96,
855 Portland, OR, pp 181–187
- 856 Werner T (2007) A linear programming approach to max-sum problem: a review. *IEEE Trans Pattern*
857 *Recognit Mach Intell* 29(7):1165–1179
- 858 Wilson N (2005) Decision diagrams for the computation of semiring valuations. In: Proceedings of
859 IJCAI-05, Edinburgh, Scotland, pp 331–336
- 860 Zytynicki M, Gaspin C, Schiex T (2008) DARN! A soft constraint solver for RNA motif localization.
861 *Constraints* 13(1–2):91–109
- 862 Zytynicki M, Gaspin C, de Givry S, Schiex T (2009) Bounds arc consistency for weighted CSPs. *J*
863 *Artif Intell Res* 35:93–621

Author Queries

Chapter 7

Query Refs.	Details Required	Author's response
AQ1	Please check and confirm if the inserted citation of Table 2 is correct. If not, please suggest an alternate citation. Please note that tables should be cited sequentially in the text.	

UNCORRECTED PROOF

MARKED PROOF

Please correct and return this set

Please use the proof correction marks shown below for all alterations and corrections. If you wish to return your proof by fax you should ensure that all amendments are written clearly in dark ink and are made well within the page margins.

<i>Instruction to printer</i>	<i>Textual mark</i>	<i>Marginal mark</i>
Leave unchanged	... under matter to remain	Ⓟ
Insert in text the matter indicated in the margin	∧	New matter followed by ∧ or ∧ [Ⓢ]
Delete	/ through single character, rule or underline or ┌───┐ through all characters to be deleted	Ⓞ or Ⓞ [Ⓢ]
Substitute character or substitute part of one or more word(s)	/ through letter or ┌───┐ through characters	new character / or new characters /
Change to italics	— under matter to be changed	↙
Change to capitals	≡ under matter to be changed	≡
Change to small capitals	≡ under matter to be changed	≡
Change to bold type	~ under matter to be changed	~
Change to bold italic	≈ under matter to be changed	≈
Change to lower case	Encircle matter to be changed	≡
Change italic to upright type	(As above)	⊕
Change bold to non-bold type	(As above)	⊖
Insert 'superior' character	/ through character or ∧ where required	Υ or Υ under character e.g. Υ or Υ
Insert 'inferior' character	(As above)	∧ over character e.g. ∧
Insert full stop	(As above)	⊙
Insert comma	(As above)	,
Insert single quotation marks	(As above)	ʹ or ʸ and/or ʹ or ʸ
Insert double quotation marks	(As above)	ʼ or ʼ and/or ʼ or ʼ
Insert hyphen	(As above)	⊥
Start new paragraph	┌	┌
No new paragraph	┐	┐
Transpose	└┐	└┐
Close up	linking ○ characters	Ⓞ
Insert or substitute space between characters or words	/ through character or ∧ where required	Υ
Reduce space between characters or words		↑