Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®
Artificial Intelligence 154 (2004) 199–227

Artificial Intelligence
www.elsevier.com/locate/artint

Arc consistency for soft constraints

Martin Cooper [a], Thomas Schiex [b,*]

[a] *IRIT-UPS, 118 Route de Narbonne, 31062 Toulouse Cedex 4, France*
[b] *INRA, Biométrie et Intelligence Artificielle, Chemin de Borde Rouge BP 27, 31326 Castanet Tolosan Cedex, France*

Received 28 January 2003; received in revised form 6 August 2003

Abstract

The notion of arc consistency plays a central role in constraint satisfaction [R. Dechter, Constraint Processing, Morgan Kaufmann, San Mateo, CA, 2003]. It is known since the introduction of valued and semi-ring constraint networks in 1995 that the notion of local consistency can be extended to constraint optimisation problems defined by soft constraint frameworks based on an idempotent cost combination operator. This excludes non-idempotent operators such as + which define problems which are very important in practical applications such as MAX-CSP, where the aim is to minimise the number of violated constraints.

In this paper, we show that using a weak additional axiom satisfied by most existing soft constraints proposals, it is possible to define a notion of *soft arc consistency* that extends the classical notion of arc consistency and this even in the case of non-idempotent cost combination operators. A polynomial time algorithm for enforcing this soft arc consistency exists and its space and time complexities are identical to that of enforcing arc consistency in CSPs when the cost combination operator is strictly monotonic (for example MAX-CSP).

A directional version of arc consistency, first introduced by M.C. Cooper [Reduction operations in fuzzy or valued constraint satisfaction, Fuzzy Sets and Systems 134 (3) (2003) 311–342] is potentially even stronger than the non-directional version, since it allows non-local propagation of penalties. We demonstrate the utility of directional arc consistency by showing that it not only solves soft constraint problems on trees, but that it also implies a form of local optimality, which we call arc irreducibility.

Keywords: Constraint satisfaction; Arc consistency; Soft constraints

* Corresponding author.
E-mail addresses: cooper@irit.fr (M. Cooper), tschiex@toulouse.inra.fr (T. Schiex).

0004-3702/$ – see front matter
doi:10.1016/j.artint.2003.09.002

1. Introduction

Compared to other combinatorial optimisation frameworks, the CSP framework is essentially characterised by the ubiquitous use of so-called local consistency properties and enforcing algorithms among which arc consistency is certainly preeminent.

The notion of local consistency can be characterised by a set of desirable properties:

- Local consistency is a relaxation of consistency, which means that for any consistent CSP there is an equivalent non-empty locally consistent CSP.
- This equivalent locally consistent CSP, which is unique, can be found in polynomial time by so-called enforcing or filtering algorithms.

Several papers have tried to extend the classical notion of arc consistency to weighted constraint frameworks. In such frameworks, the aim is to find an assignment that minimises combined violations. The first work in this direction is probably [19] which defined arc consistency filtering for conjunctive (max-min) fuzzy CSP.

This extension was rather straightforward and one might be tempted to think that this would be the case for other frameworks such as MAX-CSP, introduced in [11,24], where the aim is to find an assignment which minimises the (weighted) number of violated constraints. This turned out not to be the case. Later works tried to extend arc consistency in a systematic way using axiomatic frameworks to characterise the properties of the operator used to combine violations:

- The Semi-Ring CSP framework was introduced in [4]. In this work, the extension of arc consistency enforcing is induced by a generalisation of the fundamental relational operators such as projection, intersection and join. The essential conclusion of this work is that extended arc consistency works as long as the operator used to combine violations is idempotent. This includes the case of conjunctive fuzzy CSP (in which we try to minimise the violation of the most violated constraint) and also some other cases with partial orders. For MAX-CSP and other related cases, the algorithm may not terminate and may also provide non-equivalent CSPs.
- The Valued CSP framework was introduced in [23]. Here, the extension of the arc consistency property is essentially based on the notion of relaxation. The same conclusion as in the Semi-Ring CSP framework was reached for idempotent operators. For other frameworks such as MAX-CSP, it was shown that the problem of checking the extended arc consistency property defines an NP-complete problem.

These two algebraic structures have been compared in [3] where it is shown that the assumption of a total order in semi-ring CSPs is enough to reduce them to Valued CSPs.

Parallel to these tentative extensions of arc consistency, other research such as [1,15,16,25] tried to provide improved lower bounds for MAX-CSP. The idea of extending arc consistency was abandoned in order to simply provide the most important service, i.e., the ability to detect that a CSP has no solution whose cost is below a given threshold.

Globally, each of these proposals violates some of the desirable properties of local consistency. In this paper we show that it is possible, by the addition to the Valued CSP framework of a single axiom, to define an extended arc consistency notion that has all the desirable properties of classical arc consistency except for the uniqueness of the arc consistency closure. It has also the pleasant property that in the idempotent operator cases, it reduces to existing working definitions and uniqueness is recovered.

It has been shown [22] that a lower bound can easily be built from any of the arc consistency closures and that this lower bound generalises and improves upon existing lower bounds [1,15,16,25]. In this paper, we also consider a directional version of arc consistency that improves lower bounds by propagating partial inconsistencies and not only value deletions as [15,16]. In fact, we show that directional arc consistency, first defined in [5], defines a locally optimal lower bound.

2. Notations and definitions

A constraint satisfaction problem (CSP) is a triple $\langle X, D, C\rangle$. X is a set of n variables $X = \{1, \ldots, n\}$. Each variable $i \in X$ has a domain of values $d_i \in D$ and can be assigned any value $a \in d_i$, also noted (i, a). d will denote the cardinality of the largest domain of a CSP. C is a set of constraints. Each constraint $c_P \in C$ is defined over a set of variables $P \subseteq X$ (called the scope of the constraint) by a subset of the Cartesian product $\prod_{i\in P} d_i$ which defines all consistent tuples of values. The cardinality $|P|$ is the arity of the constraint c_P. r will denote the largest arity of a CSP. We assume, without loss of generality, that at most one constraint is defined over a given set of variables. The set C is partitioned into two sets $C = C^1 \cup C^+$ where C^1 contains all unary constraints. For simplification, the unary constraint on variable i will be denoted c_i, binary constraints being denoted c_{ij}. $e = |C|$ will denote the number of constraints in a CSP. If $J \subseteq X$ is a set of variables, then $\ell(J)$ denotes the set of all possible labellings for J, i.e., the Cartesian product $\prod_{i\in J} d_i$ of the domains of the variables in J. Let $P \subset X$, and $t \in \ell(P)$, the projection of the tuple t onto a set of variables $V \subseteq X$ is denoted by $t_{\downarrow V}$. If $i \in P$, $a \in d_i$ and $t \in \ell(P - \{i\})$, we denote by $t \cdot a$ the tuple u of $\ell(P)$ such that $u_{\downarrow\{i\}} = a$ and $u_{\downarrow P-\{i\}} = t$. A tuple of values t satisfies a constraint c_P if $t_{\downarrow P} \in c_P$. Finally, a tuple $t \in \ell(X)$ is a solution iff it satisfies all the constraints in C.

3. Valued CSP

Valued CSPs (or VCSPs) were initially introduced in [23]. A valued CSP is obtained by associating a valuation with each constraint. The set E of all possible valuations is assumed to be totally ordered and its maximum element is used to represent total inconsistency. When a tuple violates a set of constraints, its valuation is computed by combining the valuations of all violated constraints using an aggregation operator, denoted by $\oplus$. This operator must satisfy a set of properties that are captured by a set of axioms defining a so-called *valuation structure*.

Definition 1. A *valuation structure* is defined as a tuple $\langle E, \oplus, \succcurlyeq \rangle$ such that:

- E is a set, whose elements are called valuations, which is totally ordered by $\succcurlyeq$, with a maximum element denoted by $\top$ and a minimum element denoted by $\bot$;
- E is closed under a binary operation $\oplus$ that satisfies:
  - $\forall \alpha, \beta \in E, (\alpha \oplus \beta) = (\beta \oplus \alpha)$; (commutativity)
  - $\forall \alpha, \beta, \gamma \in E, (\alpha \oplus (\beta \oplus \gamma)) = ((\alpha \oplus \beta) \oplus \gamma)$; (associativity)
  - $\forall \alpha, \beta, \gamma \in E, (\alpha \succcurlyeq \beta) \Rightarrow ((\alpha \oplus \gamma) \succcurlyeq (\beta \oplus \gamma))$; (monotonicity)
  - $\forall \alpha \in E, (\alpha \oplus \bot) = \alpha$; (neutral element)
  - $\forall \alpha \in E, (\alpha \oplus \top) = \top$. (annihilator)

From an algebraical point of view, this structure can be described as a positive totally ordered commutative monoid, also known as a positive *tomonoid*. According to [9], even the simplest questions on the general structure of tomonoids are still open. When E is restricted to $[0, 1]$, this structure is also known in uncertain reasoning, as a triangular co-norm [12].

It is now possible to define valued CSPs. Note that, for the sake of generality, rather than considering that a valuation is associated with each constraint, as in [23], we consider that a valuation is associated with each tuple of each constraint. As observed in [3], the two approaches are essentially equivalent.

Definition 2. A valued CSP is a tuple $\langle X, D, C, S \rangle$ where X is a set of n variables $X = \{1, \ldots, n\}$, each variable $i \in X$ has a domain of possible values $d_i \in D$. $C = C^1 \cup C^+$ is a set of constraints and $S = \langle E, \oplus, \succcurlyeq \rangle$ is a valuation structure. Each constraint $c_P \in C$ is defined over a set of variables $P \subseteq X$ as a function $c_P : \prod_{i \in P} d_i \to E$.

An assignment t of values to some variables $J \subseteq X$ can be simply evaluated by combining, for all assigned constraints c_P (i.e., such that $P \subseteq J$), the valuations of the projection of the tuple t on P:

Definition 3. In a VCSP $V = \langle X, D, C, S \rangle$, the valuation of an assignment t to a set of variables $J \subseteq X$ is defined by:

$$\mathcal{V}_V(t) = \bigoplus_{c_P \in C, P \subseteq J} \big[c_P(t_{\downarrow P})\big].$$

The problem usually considered is to find a complete assignment with a minimum valuation. Globally, the semantics of a VCSP is defined by the valuations $\mathcal{V}(t)$ of assignments t to X.

The choice of axioms is quite natural and is usual in the field of uncertain reasoning. The ordered set E simply allows us to express different degrees of constraint violation. The commutativity and associativity guarantee that the valuation of an assignment is independent of the order in which valuations are combined. The monotonicity of $\oplus$ guarantees that assignment valuations cannot decrease when constraint violations increase. For a more detailed analysis and justification of the VCSP axioms, we invite the reader to consult [23] which also emphasise the difference between idempotent and strictly monotonic aggregation operators $\oplus$.

Definition 4. An operator $\oplus$ is idempotent if $\forall \alpha \in E$, $(\alpha \oplus \alpha) = \alpha$. It is strictly monotonic if $\forall \alpha, \beta, \gamma \in E$, $(\alpha \succ \beta) \wedge (\gamma \neq \top) \Rightarrow (\alpha \oplus \gamma) \succ (\beta \oplus \gamma)$.

As shown in [23], these two properties are incompatible as soon as $|E| > 2$. The only valuation structures with an idempotent operator correspond to classical and possibilistic CSP [21] (min-max dual to the conjunctive fuzzy CSP framework) which use $\oplus = \max$ as the aggregation operator. Other soft CSP frameworks such as MAX-CSP, lexicographic CSP or probabilistic CSP use a strictly monotonic operator.

Arc consistency enforcing must yield an equivalent problem, the so-called arc-consistency closure. Several notions of equivalence were introduced in [7,23] that enabled us to compare pairs of VCSPs with different valuations structure. In this paper, the notion of equivalence will only be used to compare pairs of VCSPs with the same valuation structure and can therefore be simplified and strengthened.

Definition 5. Two VCSPs $V = \langle X, D, C, S \rangle$ and $V' = \langle X, D, C', S \rangle$ are equivalent iff for all complete assignments t to X, we have:

$$\mathcal{V}_V(t) = \mathcal{V}_{V'}(t).$$

4. Fair valuation structures

We start with an introductory example. In the remainder of the paper, in order to illustrate the notions introduced on concrete examples, we will consider binary weighted MAX-CSPs which correspond to valued CSPs using the strictly monotonic valuation structure $\langle \mathbb{N} \cup \{\infty\}, +, \geqslant \rangle$. To describe such problems, we use an undirected graph representation where vertices represent values. For all pairs of variables $i, j \in X$ such that $c_{ij} \in C$, for all values $a \in d_i$, $b \in d_j$ such that $c_{ij}(a, b) \neq \bot = 0$, an edge connects the values (i, a) and (j, b). The weight of this edge is set to $c_{ij}(a, b)$. Unary constraints are represented by weights associated with vertices, weights equal to 0 being omitted.

Let us consider the weighted MAX-CSP in Fig. 1(a). It has two variables numbered 1 and 2, each with two values a and b together with a single constraint. The constraint forbids pair $((1, b), (2, b))$ with cost 1 and forbids pairs $((1, a), (2, a))$ and $((1, b), (2, a))$ completely (with cost ∞). The pair $((1, a), (2, b))$ is completely authorised and the corresponding edge is therefore omitted.

If we assign the value b to variable 1, it is known for sure that a cost of 1 must be paid since all extensions of $(1, b)$ to variable 2 incur a cost of at least 1. Projecting this minimum cost down from c_{12} would make this explicit and induce a unary constraint on 1 that forbids $(1, b)$ with cost 1. However if we simply add this constraint to the MAX-CSP, as was proposed in [4] for problems with an idempotent operator, the resulting CSP is not equivalent. The complete assignment $((1, b), (2, b))$ which initially had a cost of 1 would now have a cost of 2. In order to preserve equivalence, we must "compensate" for the induced unary constraint. This can be done by simply subtracting 1 from all the tuples that contain the value $(1, b)$. The corresponding equivalent CSP is shown in Fig. 1(b): the edge $((1, b), (2, b))$ of cost 1 has disappeared (the associated weight is now 0) while the edge

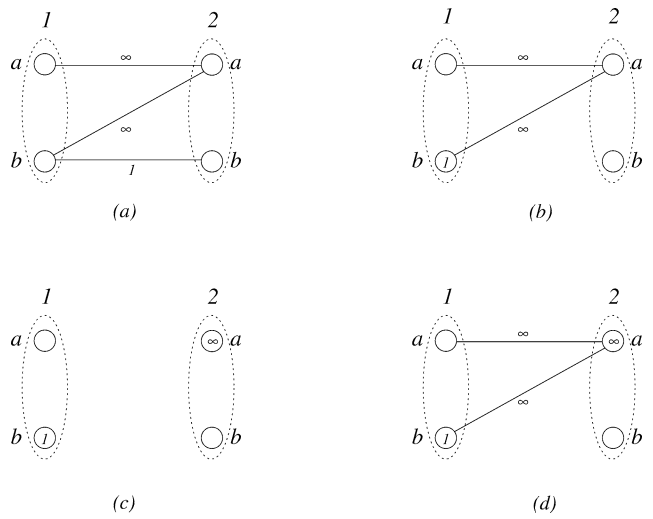

Fig. 1. Four equivalent instances of MAX-CSP.

$((1, b), (2, a))$ is unaffected since it has infinite weight and $\infty \ominus 1 = \infty$. We can repeat this process for variable 2: all extensions of value $(2, a)$ have infinite cost. Thus we can add a unary constraint that completely forbids value $(2, a)$. In this specific case, and because the valuation ∞ satisfies $\infty \oplus \infty = \infty$, we can either compensate for this (Fig. 1(c)) or not (Fig. 1(d)). In both cases, an equivalent MAX-CSP is obtained. Between the problems in Figs. 1(c) and 1(d), we prefer the problem in Fig. 1(d) because it makes information explicit both at the domain and constraint level.

This type of projection mechanism underlies most of the lower bounds defined for MAX-CSP [1,15,16,25]. To our knowledge, the introduction of a "compensation" mechanism for the definition and establishment of arc consistency appeared in [22]. It was previously introduced in [13] on MAX-CSP, independently of any notion of arc consistency.

Suppose now that the problem in Fig. 1 is part of an instance of MAX-CSP on four variables, as shown in Fig. 2(a). As in crisp CSP, inconsistencies can propagate from domains up to constraints. The cost of ∞ for $(2, a)$ can be duplicated in the costs of the pairs $((2, a), (3, a))$ and $((2, a), (3, b))$. Since $c_{23}(b, b) = \infty$, this in turn implies that the assignment $(b, 3)$ inevitably has a cost of ∞. No further propagation of infinite costs can be performed.

A similar process can be applied to finite costs but one must take care to compensate any cost change. The cost 1 of $(1, b)$ can be first shifted to the constraint c_{01}: the costs of the pairs $((0, a), (1, b))$ and $((0, b), (1, b))$ become equal to 1 and the cost of the value $(1, b)$ is set to 0. Since now $c_{01}(a, a) = c_{01}(a, b) = 1$, a cost of 1 can be projected onto value $(0, a)$. Fig. 2(b) shows the result of such propagations. In the case of finite costs, the process is obviously not terminated since one could forever shift this cost back and forth between values $(0, a)$ and $(1, b)$.

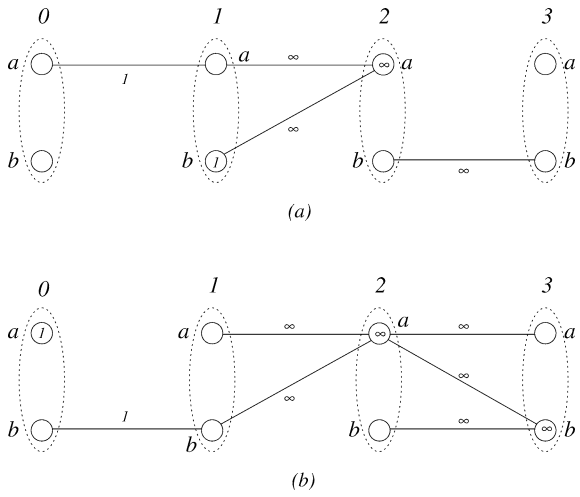

Fig. 2. Two equivalent instances of MAX-CSP.

4.1. A new axiom for VCSPs

To formalise and generalise the ideas presented in the previous section to other valuation structures, we have to be able to compensate for the information added by projecting weights down onto domains. This is made possible by the following additional definition:

Definition 6. In a valuation structure $S = \langle E, \oplus, \succcurlyeq \rangle$, if $\alpha, \beta \in E$, $\alpha \preccurlyeq \beta$ and there exists a valuation $\gamma \in E$ such that $\alpha \oplus \gamma = \beta$, then γ is known as a difference of β and α.

The valuation structure S is *fair* if for any pair of valuations $\alpha, \beta \in E$, with $\alpha \preccurlyeq \beta$, there exists a maximum difference of β and α. This unique maximum difference of β and α is denoted by $\beta \ominus \alpha$.

Lemma 7. *Let $S = \langle E, \oplus, \succcurlyeq \rangle$ be a fair valuation structure. Then $\forall u, v, w \in E$, $w \preccurlyeq v$, we have $(v \ominus w) \preccurlyeq v$ and $(u \oplus w) \oplus (v \ominus w) = (u \oplus v)$.*

Proof. By definition, $(v \ominus w) \oplus w = v$. From the monotonicity of $\oplus$, this proves that $(v \ominus w) \preccurlyeq v$ (this inequality becomes strict if $\oplus$ is strictly monotonic and $v \neq \top$). The second property follows from the commutativity and associativity of $\oplus$: we have $(u \oplus w) \oplus (v \ominus w) = u \oplus ((v \ominus w) \oplus w) = (u \oplus v)$. □

Most existing concrete soft constraint frameworks, including all those with either an idempotent or strictly monotonic operator $\oplus$ are fair.

Example 8. If $\oplus$ is idempotent, then it can easily be shown that $\oplus = \max$ [23]. When $\oplus = \max$, we have $\ominus = \max$, since $\max(\max(\alpha, \beta), \alpha) = \beta$ whenever $\alpha \preccurlyeq \beta$. When $\alpha = \beta$, then any valuation $\gamma \prec \alpha$ is also a valid difference of β and α but it is clearly not maximum. Note that classical CSPs can be defined as VCSPs over the idempotent valuation structure

$S = \langle \{\bot, \top\}, \max, \succcurlyeq \rangle$, where $\bot$ represents *true* and $\top$ represents *false*. The operator $\oplus$ is also idempotent in possibilistic CSPs [21] which define a min-max problem which is dual to the max-min problem of conjunctive fuzzy CSPs [5,19].

Example 9. In the strictly monotonic valuation structure $\langle \mathbb{N} \cup \{\infty\}, +, \geqslant \rangle$, $\ominus$ is defined by $\beta \ominus \alpha = \beta - \alpha$ for finite valuations $\alpha, \beta \in \mathbb{N}$, $\alpha \leqslant \beta$ and $(\infty \ominus \alpha) = \infty$ for all $\alpha \in \mathbb{N} \cup \{\infty\}$. It can be shown similarly that Probabilistic CSP [23] are also fair.

In the general case of any strictly monotonic operator $\oplus$, the difference operator may not exist in E, but it has been proved in [5] that the difference operator can always be constructed by embedding the valuation structure in a larger valuation structure derived from the set $E \times E$, where (β, α) represents the imaginary $\beta \ominus \alpha$. This can be compared with embedding $\mathbb{R}$ in $\mathbb{C}$ so as to allow us to take square roots of negative numbers. This construction is simple and effective. It can for example be used on lexicographic CSPs [10] for which differences are not always defined in the original valuation structure. Note that another possible approach is to transform the lexicographic CSP into a VCSP on the valuation structure $\langle \mathbb{N} \cup \{\infty\}, +, \geqslant \rangle$ using the simple transformation described in [23].

4.2. Equivalence-preserving transformations

As it has been demonstrated in the examples of Figs. 1 and 2, it is possible to transform a MAX-CSP into an equivalent but different MAX-CSP using local transformations (involving only one non-unary constraint). Such operations will be called equivalence-preserving transformations:

Definition 10. The *subproblem* of a VCSP $V = \langle X, D, C, S \rangle$ on $J \subseteq X$ is the VCSP $V(J) = \langle J, D_J, C_J, S \rangle$, where $D_J = \{d_j : j \in J\}$ and $C_J = \{c_P \in C : P \subseteq J\}$.

Definition 11. For a VCSP V, an *equivalence-preserving transformation* of V on $J \subseteq X$ is an operation which transforms the subproblem of V on J into an equivalent VCSP. If $C_J = \{c_P \in C : P \subseteq J\}$ contains only one non-unary constraint, such an operation is called an *equivalence-preserving arc transformation*.

Example 12. The procedures Project and Extend described in Algorithm 1 are examples of equivalence-preserving transformations.

Project transforms a VCSP by shifting valuations from the tuples of a given non-unary constraint c_P to the value (i, a), where $i \in P$, $a \in d_i$. In order to preserve equivalence, any increase at the unary level is compensated at the tuple level (line 1 of Algorithm 1).

Conversely, Extend shifts the valuation from value (i, a) to the tuples of the constraint c_P where $i \in P$. Again, the fairness of the valuation structure allows us to compensate for the possible increase of the tuple valuations by an operation at the unary level (line 2 of Algorithm 1).

Theorem 13. *Given any fair VCSP $V = \langle X, D, C, S \rangle$, for any $c_P \in C^+$, $i \in P$, $a \in d_i$, the application of* Project *or* Extend *on V yields an equivalent VCSP.*

Procedure Project(c_P, i, a)
  $\beta \leftarrow \min_{t \in \ell(P-\{i\})}(c_P(t \cdot a))$
  $c_i(a) \leftarrow c_i(a) \oplus \beta$
  foreach ($t \in \ell(P - \{i\})$) **do**
1 $c_P(t \cdot a) \leftarrow c_P(t \cdot a) \ominus \beta$

Procedure Extend(i, a, c_P)
  foreach ($t \in \ell(P - \{i\})$) **do**
    $c_P(t \cdot a) \leftarrow c_P(t \cdot a) \oplus c_i(a)$
2 $c_i(a) \leftarrow c_i(a) \ominus c_i(a)$

Algorithm 1. Two basic equivalence-preserving arc-transformations.

Proof. To demonstrate equivalence, it is sufficient to prove that the value of $c_P(t \cdot a) \oplus c_i(a)$ is an invariant of Project(c_P, i, a) and Extend(i, a, c_P). For any $t \in \ell(P - \{i\})$, let γ be the initial value of $c_P(t \cdot a)$ and δ the initial value of $c_i(a)$. After the execution of Project, we have $(c_P(t \cdot a) \oplus c_i(a)) = (\gamma \ominus \beta) \oplus (\delta \oplus \beta) = \gamma \oplus \delta$. After the execution of Extend, we have $(c_P(t \cdot a) \oplus c_i(a)) = (\gamma \oplus \delta) \oplus (\delta \ominus \delta) = \gamma \oplus \delta$. This proves the invariances. □

As the example of Fig. 2 showed in the case of MAX-CSP, the iterated application of equivalence-preserving transformations such as Project and Extend does not necessarily lead to a quiescent state. The termination conditions for a general class of soft constraint propagation algorithms have been analyzed in detail in [2]. However, these results do not apply here because the equivalence-preserving functions above do not satisfy all the required properties.[1] The two following sections show how a controlled application of carefully designed equivalence-preserving transformations can guarantee that a quiescent state will always be reached.

5. Soft arc consistency

In classical CSPs and in existing soft constraint propagation algorithms, arc consistency enforcing always increases the information available. In the case of soft arc consistency, application of arc transformations will be limited to operations that either increase the information available at the variable level even if they *lower the information available at larger arity level* or that increase information available at the constraint level *as long as they do not lower the information available at the variable level*. In the next section, we try to better characterise when this is possible.

5.1. On the structure of valuation structures

Definition 14. In a valuation structure $\langle E, \oplus, \succcurlyeq \rangle$, an element $\alpha \in E$ is an idempotent element iff $\alpha \oplus \alpha = \alpha$.

[1] More specifically, the functions Project and Extend are not *inflationary* with respect to the order $\sqsubseteq_P$ as defined in [2].

Idempotent elements can be duplicated without affecting valuations. They can be propagated, in the same way as inconsistencies are in crisp CSPs. Non-idempotent elements α can be shifted from one constraint to another, but each addition of α must be compensated by a subtraction elsewhere.

In a valuation structure $\langle E, \oplus, \succcurlyeq \rangle$, if $\oplus$ is idempotent then all elements of E are idempotent. If $\oplus$ is a strictly monotonic operator then the only idempotent elements are $\bot$ and $\top$. Intermediate cases occur in the following examples:

Example 15. Imagine the possible sentences for driving offences. Suppose that penalty points (up to a maximum of 12) are awarded for minor offences, whereas serious offences are penalised by suspension of the offender's driving licence for a period of y years, for some positive integer y. A driver who accumulates 12 penalty points receives an automatic one-year suspension of his/her license. The set of sentences can be modelled by a valuation structure $S = \langle E, \oplus, \succcurlyeq \rangle$ of the form:

$$
\begin{aligned}
&E = \big\{(p, 0):\ p \in \{0, \ldots, 12\}\big\} \cup \big\{(0, y):\ y \in \mathbb{N}^* \cup \{\infty\}\big\},\\
&(p, y) \prec (p', y') \quad \Leftrightarrow \quad (y < y') \vee \big((y = y' = 0) \wedge (p < p')\big),\\
&(p, 0) \oplus (p', 0) = \big(\min(p + p', 12), 0\big),\\
&(p, y) \oplus (p', y') = (0, y + y') \quad \text{if } (y + y' \neq 0).
\end{aligned}
$$

Note that $(12, 0) \prec (0, 1)$ even though they both give rise to a one-year license suspension. The penalty $(0, 1)$ is deemed to be worse because it can be cumulated. For example $(0, 1) \oplus (0, 1) = (0, 2)$, whereas $(12, 0) \oplus (12, 0) = (12, 0)$. Apart from $\bot = (0, 0)$ and $\top = (0, \infty)$, this valuation structure contains another idempotent valuation, namely $(12, 0)$. This is a fair valuation structure since $\oplus$ has the following inverse operation $\ominus$:

$$
\begin{aligned}
&(p, 0) \ominus (p', 0) = (p - p', 0) \quad \text{if } p < 12,\\
&(12, 0) \ominus (p', 0) = (12, 0),\\
&(0, y) \ominus (p', y') = (0, y - y').
\end{aligned}
$$

Example 16. Another interesting case occurs if, for example, a company wants to minimise both financial loss F and loss of human life H if a fire should break out in its factory. Supposing that the company considers that no price can be put on human life, we must have

$$(F, H) < (F', H') \quad \Leftrightarrow \quad (H < H') \vee (H = H' \wedge F < F').$$

If a financial loss of $F_{\max}$ represents bankruptcy, then

$$(F, H) \oplus (F', H') = \big(\min\{F + F', F_{\max}\}, H + H'\big)$$

and $(F_{\max}, 0)$ is an idempotent element which is strictly less than $\top$. Note that this valuation structure is not fair, since it is impossible to define $\alpha = (0, 1) \ominus (F_{\max}, 0)$ such that $\alpha \oplus (F_{\max}, 0) = (0, 1)$.

Example 17. Consider a valuation structure $S = \langle \mathbb{N} \cup \{\infty, \top\}, \oplus, \geqslant \rangle$ composed of prison sentences. Sentences may be of n years, life imprisonment (represented by ∞) or the

death penalty (represented by $\top$). There is a rule that states that two life sentences lead automatically to a death sentence: in other words $(\infty \oplus \infty) = \top$. Otherwise, sentences are cumulated in the obvious way: $\forall m, n \in \mathbb{N}$ $(m \oplus n = m + n)$; $\forall n \in \mathbb{N}$ $(\infty + n = \infty)$; $\forall \alpha \in E$ $(\top \oplus \alpha = \top)$. Although every pair $\beta, \alpha \in E$, $\alpha \leqslant \beta$ possesses a difference, this valuation structure is not fair since the set of differences of ∞ and ∞ is $\mathbb{N}$ and hence no *maximum* difference of ∞ and ∞ exists. However, S can easily be rendered fair by replacing $\mathbb{N}$ by $\{0, 1, 2, \ldots, 150\}$, for example.

The following results show that all fair valuation structures are composed of slices separated by idempotent values, each slice being independent of the others.

Lemma 18. *Let $S = \langle E, \oplus, \succcurlyeq \rangle$ be a valuation structure. If $\alpha, \beta \in E$, α is an idempotent element and $\beta \preccurlyeq \alpha$ then $\alpha \oplus \beta = \alpha$. If S is fair, then $\alpha \ominus \beta = \alpha$.*

Proof. Since $\beta \preccurlyeq \alpha$, it follows that $\alpha \preccurlyeq \alpha \oplus \beta \preccurlyeq \alpha \oplus \alpha = \alpha$, by monotonicity. Thus, $\alpha \oplus \beta = \alpha$. Furthermore, $\alpha \oplus \beta = \alpha$ shows that α is a difference of α and β. It is the maximum difference since $\alpha \ominus \beta = (\alpha \ominus \beta) \oplus \bot \preccurlyeq (\alpha \ominus \beta) \oplus \beta = \alpha$, by monotonicity. Thus $\alpha \ominus \beta = \alpha$. □

Lemma 19. *Let $S = \langle E, \oplus, \succcurlyeq \rangle$ be a fair valuation structure. If $\alpha, \beta \in E$, α is an idempotent element and $\beta \succcurlyeq \alpha$ then $\alpha \oplus \beta = \beta$ and $\beta \ominus \alpha = \beta$.*

Proof. Since α is idempotent, $\beta \oplus \alpha = (\beta \ominus \alpha) \oplus \alpha \oplus \alpha = (\beta \ominus \alpha) \oplus \alpha = \beta$. Furthermore, this shows that β is a difference of β and α. It is the maximum difference since $\beta \ominus \alpha \preccurlyeq \beta$, by Lemma 7. Thus, $\beta \ominus \alpha = \beta$. □

Theorem 20 (Slice Independence Theorem). *Let $S = \langle E, \oplus, \succcurlyeq \rangle$ be a fair valuation structure. Let $\beta, \gamma \in E$, $\beta \preccurlyeq \gamma$, and let $\alpha_0, \alpha_1 \in E$ be idempotent valuations such that $\alpha_0 \preccurlyeq \gamma \preccurlyeq \alpha_1$. Then $\alpha_0 \preccurlyeq (\gamma \oplus \beta) \preccurlyeq \alpha_1$ and $\alpha_0 \preccurlyeq (\gamma \ominus \beta) \preccurlyeq \alpha_1$.*

Proof. By monotonicity, $\beta \oplus \gamma \preccurlyeq \alpha_1 \oplus \gamma = \alpha_1$ by Lemma 18. By Lemma 19, $\gamma = \gamma \oplus \alpha_0 = (\gamma \ominus \beta) \oplus \beta \oplus \alpha_0 = (\gamma \ominus \beta) \oplus \alpha_0 \oplus \beta$. Therefore, $((\gamma \ominus \beta) \oplus \alpha_0)$ is a difference of γ and β. Since $\gamma \ominus \beta$ is a maximum difference, $\gamma \ominus \beta \succcurlyeq (\gamma \ominus \beta) \oplus \alpha_0 \succcurlyeq \alpha_0$, by monotonicity. The remaining equalities follow from monotonicity. □

The following results will be useful for the proof of correctness of the arc consistency enforcing algorithm given in the next section.

Theorem 21. *Let $S = \langle E, \oplus, \succcurlyeq \rangle$ be a fair valuation structure. For all $\alpha \in E$, $\alpha \ominus \alpha$ is the maximum idempotent valuation less than or equal to α.*

Proof. Let $\beta = \alpha \ominus \alpha$. Now $\alpha \oplus (\beta \oplus \beta) = (\alpha \oplus \beta) \oplus \beta = \alpha \oplus \beta = \alpha$, which shows that $\beta \oplus \beta$ is a difference of α and α. By Definition 6, β is the maximum difference. Therefore, $\beta \succcurlyeq \beta \oplus \beta$. Since $\beta \preccurlyeq \beta \oplus \beta$ by monotonicity, we have $\beta = \beta \oplus \beta$ and hence β

is idempotent. Maximality follows from Theorem 20, since for all idempotent valuations $\alpha_0 \preccurlyeq \alpha$, we have $\alpha_0 \preccurlyeq (\alpha \ominus \alpha)$. □

Lemma 22. *Let $S = \langle E, \oplus, \succcurlyeq \rangle$ be a fair valuation structure. For all $\alpha, \beta \in E$, $((\alpha \oplus \beta) \ominus \alpha) \ominus \beta = (\alpha \oplus \beta) \ominus (\alpha \oplus \beta)$.*

Proof. Let $\gamma = (\alpha \oplus \beta) \ominus (\alpha \oplus \beta)$. By Theorem 21, γ is idempotent. Now $\gamma \preccurlyeq (\alpha \oplus \beta)$, by Lemma 7. Let $\delta = ((\alpha \oplus \beta) \ominus \alpha) \ominus \beta$. Then the fact that $\delta \succcurlyeq \gamma$ follows from two applications of the Slice Independence Theorem. But $\delta \oplus (\alpha \oplus \beta) = (\alpha \oplus \beta)$. Therefore δ is a difference of $(\alpha \oplus \beta)$ and $(\alpha \oplus \beta)$. γ being the maximum difference, this shows that $\delta \preccurlyeq \gamma$ and $\delta = \gamma$. □

Theorem 23. *Let $S = \langle E, \oplus, \succcurlyeq \rangle$ be a fair valuation structure. For all $\alpha, \beta \in E$, either $(\alpha \oplus \beta) \ominus \beta = \alpha$ or $(\alpha \oplus \beta) \ominus \beta = (\alpha \oplus \beta) \ominus (\alpha \oplus \beta)$ which is idempotent and strictly greater than α.*

Proof. Let $\gamma = (\alpha \oplus \beta) \ominus (\alpha \oplus \beta)$. Now $(\alpha \oplus \beta) \ominus \beta = (((\alpha \oplus \beta) \ominus \beta) \ominus \alpha) \oplus \alpha = \gamma \oplus \alpha$, by Lemma 22. Since γ is idempotent, $\gamma \oplus \alpha$ equals either α (if $\alpha \succcurlyeq \gamma$, Lemma 19) or γ (if $\gamma \succcurlyeq \alpha$, Lemma 18). In this case, the fact that γ is idempotent follows directly from Theorem 21. □

The following result will be useful for bounding the complexity of enforcing arc consistency on arbitrary fair VCSP:

Theorem 24. *Any VCSP $V = \langle X, D, C, S \rangle$ on a fair valuation structure $S = \langle E, \oplus, \succcurlyeq \rangle$ is equivalent to a VCSP on a valuation structure S' with no more than $2ed^r + 2$ idempotent valuations.*

Proof. For any $\beta \in E$ define

$$Slice(\beta) = \{\alpha \in E\colon \nexists \gamma \text{ idempotent in } E \text{ s.t. } (\alpha \prec \gamma \preccurlyeq \beta) \vee (\alpha \succ \gamma \succcurlyeq \beta)\}.$$

If β is idempotent then $Slice(\beta) = \{\beta\}$, otherwise $Slice(\beta)$ is the set of valuations α for which there is no intermediate idempotent valuation γ lying between α and β. Each $Slice(\beta)$ contains at most two idempotent valuations, namely the maximum idempotent valuation less than or equal to β (which is in fact $\beta \ominus \beta$, by Theorem 21) and the minimum idempotent valuation greater than or equal to β (which may or may not exist).

Let E_0 be the set of valuations taken on by the cost functions $c_P \in C$ in the VCSP and let

$$E' = \{\bot, \top\} \cup \left(\bigcup_{\beta \in E_0} Slice(\beta) \right).$$

Clearly E' contains at most $2ed^r + 2$ idempotent valuations. It is sufficient to show that E' is closed under $\oplus$ and $\ominus$.

Consider $\alpha, \beta \in E'$ such that $\alpha \preccurlyeq \beta$ and $\beta \in Slice(\beta_0)$ for some $\beta_0 \in E_0$.

Suppose that $(\alpha \oplus \beta) \notin E'$. This implies that $\exists \gamma$ idempotent in E such that $\alpha \oplus \beta \succ \gamma \succcurlyeq \beta_0$. But, since $\beta \preccurlyeq \gamma$ by the definition of $Slice(\beta_0)$, this contradicts Theorem 20. Similarly, $\beta \ominus \alpha \notin E'$ implies that $\exists \gamma$ idempotent in E such that $\beta \ominus \alpha \prec \gamma \preccurlyeq \beta_0$. But, since $\beta \succcurlyeq \gamma$ by the definition of $Slice(\beta_0)$, this again contradicts Theorem 20. □

By Theorem 24, we can now assume, without loss of generality, that the number of idempotent valuations in the valuation structure is finite.

5.2. A definition of soft arc consistency

Before giving an arc consistency enforcing algorithm which is valid over any fair valuation structure, we require a formal definition of arc consistency for fair VCSPs. We first consider the usual restriction to binary VCSPs.

Definition 25. A fair binary VCSP is *arc consistent* if for all $i, j \in X$ such that $c_{ij} \in C^+$, for all $a \in d_i$ we have:

(1) $\forall b \in d_j$,

$$c_{ij}(a,b) = \big(c_i(a) \oplus c_{ij}(a,b) \oplus c_j(b)\big) \ominus \big(c_i(a) \oplus c_j(b)\big);$$

(2) $c_i(a) = \min_{b \in d_j}(c_i(a) \oplus c_{ij}(a,b))$.

Condition (1) states that $c_{ij}(a,b)$ has been increased to the maximum element in E which does not increase the valuation $(c_i(a) \oplus c_{ij}(a,b) \oplus c_j(b))$ of (a,b) on $\{i,j\}$. If $\oplus$ is strictly monotonic or idempotent, then this is equivalent to saying that idempotent valuations have been propagated from $c_i(a)$ to $c_{ij}(a,b)$. Condition (2) says that we have propagated as much weight as possible from the constraint c_{ij} onto c_i.

To gain a better understanding of condition (1) of Definition 25 in the most general case, consider a simple valuation structure in which penalties lies in the range $\{0,1,2,3,4,5\}$, $0 = \bot$, $5 = \top$ and $\forall \alpha, \beta \in E$ $(\alpha \oplus \beta = \min(5, \alpha + \beta))$. 5 verifies $5 \ominus \alpha = 5$ for all $\alpha \preccurlyeq 5$. This non-idempotent, non-strictly monotonic valuation structure is well suited to tackle MAX-CSPs when an upper bound (here 5) on the cost of the optimal solution is known (see [14]).

Fig. 3(a) shows a 2-variable VCSP over this valuation structure. Figure 3(b) shows the result of enforcing condition (1) of Definition 25: $c_{12}(a,a)$ and $c_{12}(b,a)$ can both be increased to 5 without changing the valuations of the solutions (a,a) and (b,a). Fig. 3(c) shows the result of then enforcing condition (2): penalties are projected down from constraints to domains, as we have seen in the example of Fig. 1.

Definition 25 can be generalised to non-binary VCSP. We call this generalised arc consistency, to be consistent with the terminology employed in the CSP literature [18].

Definition 26. A fair VCSP is *generalised arc consistent* if for all $c_P \in C^+$, we have:

(1) $\forall t \in \ell(P)$, $c_P(t) = (c_P(t) \oplus \beta) \ominus \beta$, where $\beta = \bigoplus_{j \in P} c_j(t_{\downarrow\{j\}})$;
(2) $\forall i \in P$, $\forall a \in d_i$, $c_i(a) = \min_{t \in \ell(P - \{i\})}(c_i(a) \oplus c_P(t \cdot a))$.

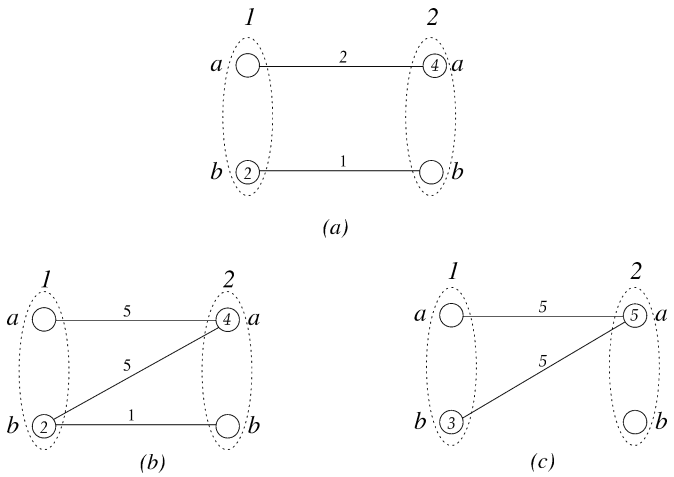

Fig. 3. (a) An example of a VCSP and how conditions (1) and (2) are enforced.

Procedure AC-Project(c_P, i, a)
  $\beta \leftarrow \min_{t \in \ell(P-\{i\})}(c_P(t \cdot a))$
  if $(c_i(a) \oplus \beta \succ c_i(a))$ **then**
    $c_i(a) \leftarrow c_i(a) \oplus \beta$
    Add $(a, \{i\}, c_i(a))$ to Q
    foreach $(t \in \ell(P - \{i\}))$ **do**
      $c_P(t \cdot a) \leftarrow c_P(t \cdot a) \ominus \beta$

Algorithm 2. Projection for soft AC enforcing.

Having given the necessary definitions, we can now define a generalised arc consistency enforcing algorithm.

5.3. Enforcing generalised arc consistency in fair VCSPs

Arc consistency is established by repeated calls to two subroutines denoted by AC-Project and AC-Extend (see Algorithms 2 and 3, respectively), called *arc consistency operations*. The data structure Q is a queue containing elements (t, P, α), where $\alpha = c_P(t)$. The subroutine AC-Project(c_P, i, a) is a simple modification of the basic equivalence-preserving transformation Project that memorises VCSP modifications in the queue for further propagation. This simple modification obviously does not alter the fact that it is an equivalence-preserving transformation.

Lemma 27. *For any fair VCSP $V = \langle X, D, C, S\rangle$, if $c_P \in C$, $i \in P$, $a \in d_i$ then the result of applying* AC-Project(c_P, i, a) *to a fair VCSP V is an equivalent VCSP in which*

$$c_i(a) = \min_{t \in \ell(P-\{i\})} \big(c_i(a) \oplus c_P(t \cdot a)\big).$$

Procedure AC-Extend(i, a, c_P)
  foreach $(t \in \ell(P - \{i\}))$ **do**
    $\beta \leftarrow \bigoplus_{j \in P} c_j((t \cdot a)_{\downarrow\{j\}})$
    $\gamma \leftarrow (c_P(t \cdot a) \oplus \beta) \ominus \beta$
    if $(\gamma \succ c_P(t \cdot a))$ **then**
      $c_P(t \cdot a) \leftarrow \gamma$
      Add $(t \cdot a, P, c_P(t \cdot a))$ to Q

Algorithm 3. Extension for soft AC enforcing.

Proof. This property follows from the fact that either $c_i(a) \oplus \beta$ is not strictly greater than $c_i(a)$ and in this case $c_i(a) = c_i(a) \oplus \min_{t \in \ell(P-\{i\})}(c_P(t \cdot a)) = \min_{t \in \ell(P-\{i\})}(c_i(a) \oplus c_P(t \cdot a))$ or else let $t_{\min}$ be the tuple such that $c_P(t_{\min}, a) = \min_{t \in \ell(P-\{i\})}(c_P(t \cdot a)) = \beta$ before execution of AC-Project. Let δ be the original value of $c_i(a)$. After execution, $c_i(a) = \delta \oplus \beta = c_i(a) \oplus c_P(t_{\min}, a)$, since AC-Project is equivalence-preserving. Therefore we have $c_i(a) \succcurlyeq \min_{t \in \ell(P-\{i\})}(c_i(a) \oplus c_P(t \cdot a))$. The equality follows from monotonicity. □

The subroutine AC-Extend(i, a, c_P) is a modified version of the equivalence-preserving transformation Extend that propagates an increase in the valuation $c_i(a)$ to all the valuations $c_P(t \cdot a)$ for $t \in \ell(P - \{i\})$ when this can be done without any compensation at the unary level. It also memorises the new valuation γ of each tuple of c_P in Q for further propagation. In this case, γ is always an idempotent valuation (by Theorem 23).

Lemma 28. *For any fair VCSP $V = \langle X, D, C, S\rangle$, if $c_P \in C$, $i \in P$, $a \in d_i$ then the result of applying* AC-Extend(i, a, c_P) *to a fair VCSP V is an equivalent VCSP in which*

$$\forall t \in \ell\big(P - \{i\}\big), \quad \big(c_P(t \cdot a) = \big(c_P(t \cdot a) \oplus \beta\big) \ominus \beta\big),$$
$$\text{where } \beta = \bigoplus_{j \in P} c_j\big((t \cdot a)_{\downarrow\{j\}}\big).$$

Proof. To demonstrate equivalence, it is sufficient to prove that $\forall t \in \ell(P - \{i\})$, $\beta \oplus c_P(t \cdot a)$ is an invariant of AC-Extend(i, a, c_P), where $\beta = \bigoplus_{j \in P} c_j((t \cdot a)_{\downarrow\{j\}})$. But this is certainly the case because, if $c_P(t \cdot a) = \alpha$ before execution of AC-Extend(i, a, c_P), then $\beta \oplus c_P(t \cdot a) = \beta \oplus ((\alpha \oplus \beta) \ominus \beta) = \beta \oplus \alpha$ after $c_P(t \cdot a)$ is updated by AC-Extend(i, a, c_P).

We know that $\alpha \preccurlyeq (\alpha \oplus \beta) \ominus \beta$, since α is clearly a difference of $\alpha \oplus \beta$ and β. If $\alpha \prec (\alpha \oplus \beta) \ominus \beta$, then $c_P(t \cdot a)$ is assigned $(\alpha \oplus \beta) \ominus \beta$. Hence, after $c_P(t \cdot a)$ is updated by AC-Extend(i, a, c_P), $\beta \oplus c_P(t \cdot a) = \beta \oplus \alpha$ and $(c_P(t \cdot a) \oplus \beta) \ominus \beta = (\alpha \oplus \beta) \ominus \beta = c_P(t \cdot a)$. □

We are now in a position to give an algorithm (Algorithm 4) for generalised arc consistency in fair VCSPs.

Theorem 29. *When* GAC *terminates, the resulting VCSP is generalised arc consistent.*

```
Procedure GAC()
    { Initialisation phase }
    foreach i ∈ X do
        foreach a ∈ d_i do
            foreach c_P ∈ C^+ s.t. i ∈ P do
                AC-Extend(i, a, c_P)
                AC-Project(c_P, i, a)
    { Propagation phase }
    while Q ≠ ∅ do
        Extract the first element (t, P, α) from Q
        if c_P(t) = α then
            if P is a singleton {i} then
                foreach c_P ∈ C^+ s.t. i ∈ P do AC-Extend(i, t_↓{i}, c_P)
            else
                foreach i ∈ P do AC-Project(c_P, i, t_↓{i})
```

Algorithm 4. Generalised arc consistency enforcing for fair VCSPs.

Proof. Consider (c_P, i, a) such that $c_P \in C$, $i \in P$, $a \in d_i$. We know that AC-Project(c_P, i, a) is called at least once during GAC, since it is called in the initialisation phase. After the last call to AC-Project(c_P, i, a)

$$c_i(a) = \min_{t \in \ell(P-\{i\})} \big(c_i(a) \oplus c_P(t \cdot a)\big)$$

by Lemma 27. This can only later become invalid by an increase in some $c_P(t)$ by AC-Extend, which would necessarily be accompanied by the addition of $(t, P, c_P(t))$ to Q and would hence entail another call of AC-Project(c_P, i, a). This contradiction demonstrates that condition (2) in Definition 26 of generalised arc consistency holds when GAC terminates.

Consider $t \in \ell(P)$ where $c_P \in C^+$. We know that AC-Extend$(i, t_{\downarrow\{i\}}, c_P)$ is called during the initialisation phase for each $i \in P$. After the last such call of AC-Extend$(i, t_{\downarrow\{i\}}, c_P)$ for any $i \in P$,

$$\forall t \in \ell(P), \quad c_P(t) = \big(c_P(t) \oplus \beta\big) \ominus \beta,$$
$$\text{where } \beta = \bigoplus_{j \in P} c_j(t_{\downarrow\{j\}})$$

by Lemma 28. This could only later become invalid by an update of $c_P(t)$ or some $c_j(t_{\downarrow\{j\}})$ by AC-Project$(c_N, j, t_{\downarrow\{j\}})$ for some $c_N \in C$ such that $j \in P \cap N$. But then a call of AC-Extend$(j, t_{\downarrow\{j\}}, c_P)$ would ensue. This contradiction shows that condition (2) of Definition 26 of generalised arc consistency also holds when GAC terminates. □

Theorem 30. GAC *has polynomial time complexity.*

Proof. Let n_1 be the number of elements (t, P, α) extracted from Q during GAC such that P is a singleton, and let n_2 be the number of elements (t, P, α) extracted from Q during GAC such that $|P| \geqslant 2$. By Theorem 24, we can assume that the valuation structure contains at most $2ed^r + 2$ idempotent valuations.

Let $c_P \in C^+$ and $t \in \ell(P)$. By Theorem 23, AC-Extend can only increase $c_P(t)$ to an idempotent valuation strictly greater than its previous valuation. AC-Project can decrease $c_P(t)$ but, by the Slice Independence Theorem, only from a non-idempotent valuation γ to a valuation larger than or equal to $\delta = \gamma \ominus \gamma$, the maximum idempotent valuation less than or equal to γ. Thus the sequence of idempotent valuations taken on by $c_P(t)$ during GAC are strictly increasing. Thus for each of the $2ed^r + 2$ attainable idempotent valuations α, (t, P, α) is added to Q at most once. Thus $n_2 \leqslant ed^r(2ed^r + 2)$.

Now n_1 cannot exceed the number of calls of AC-Project during GAC since tuples (t, P, α) such that P is a singleton are only added to Q by AC-Project. The number of calls of AC-Project is clearly bounded above by $erd + n_2 r$. Thus $n_1 + n_2 \leqslant erd + (r+1)n_2 = \mathrm{O}(e^2 d^{2r})$. Thus the total number of iterations of the while loop in GAC is a polynomial function of e and d. The results follows immediately. □

Definition 31. An arc consistent closure of a VCSP V is a VCSP which is arc consistent and which can be obtained from V by a finite sequence of applications of arc consistency operations AC-Extend and AC-Project.

Note that confluence of arc consistency enforcing is lost and therefore the arc consistent closure of a problem is not necessarily unique as it is in classical CSPs. Fig. 4(a) shows a 2-variable VCSP on the valuation structure $\langle \mathbb{N} \cup \{\infty\}, +, \geqslant \rangle$. Each edge has a weight of 1. Figs. 4(b) and 4(c) show two different arc consistency closures of this VCSP.

5.4. Maximum arc consistency

One of the practical uses of arc consistency in VCSPs is the computation of lower bounds on the valuation of an optimal solution. Obviously, given a VCSP V, the following valuation $f_{\min}(V)$ is always a lower bound on the cost of an optimal solution:

$$f_{\min}(V) = \bigoplus_{c_P \in C} \big[\min_{t \in \ell(P)} c_P(t) \big].$$

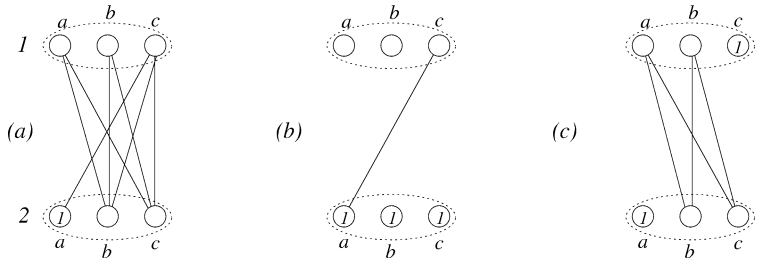

Fig. 4. A MAX-CSP and two different equivalent arc consistent closures.

From this point of view, the closure in Fig. 4(b) is preferable to the closure in Fig. 4(c) since it makes explicit the fact that 1 is a lower bound on the valuation of all solutions.

Definition 32. A VCSP V is said to be *maximum arc consistent* if it is arc consistent and if the associated lower bound $f_{\min}(V)$ is maximum over all arc consistent closures of V.

The problem of enforcing maximum arc consistency is certainly practically important and some closely related problems are known to be NP-hard [20,23]. Consider the following problem:

Problem 33 (*Max-AC*). Given a VCSP $V = (X, D, C, S)$ and a valuation $\kappa \in S$, does there exist an arc consistency closure V' of V such that $f_{\min}(V') \succcurlyeq \kappa$?

Theorem 34. *The decision problem* MAX-AC *is* NP*-hard.*

Proof. We prove this by constructing a polynomial reduction from 3-SAT to MAX-AC.

Let $I_{3\text{-SAT}}$ be an instance of 3-SAT, consisting of n variables and d clauses, each clause being the disjunction of exactly 3 literals.

We assume that each boolean variable u and its negation $\neg u$ occur exactly the same number of times in $I_{3\text{-SAT}}$. Note that if this is not initially the case, it can easily be achieved by adding the required number of tautological clauses of the form $(u \vee u \vee \neg u)$ or $(u \vee \neg u \vee \neg u)$. We will now construct an instance V of MAX-CSP on $4d$ variables such that V has an arc consistency closure V' with $f_{\min}(V') \succcurlyeq 5d/2$ iff $I_{3\text{-SAT}}$ is satisfiable. Note that d is necessarily even by our assumption that each variable u and its negation occur the same number of times.

Suppose that the boolean variable u (and its negation $\neg u$) occur in exactly $m(u)$ clauses in $I_{3\text{-SAT}}$. Then we add the gadget $G(u)$ shown in Fig. 5 to V, containing the $2m(u)$ variables $i(u, r)$, $i(\neg u, r)$ $(r = 1, \ldots, m(u))$, each with domain-size 3. Each edge joining value a at variable i and value b at variable j represents a penalty of 1, i.e., $c_{ij}(a, b) = 1$. For each clause c in $I_{3\text{-SAT}}$, we add the gadget $H(c)$ shown in Fig. 6, involving a new variable $i(c)$ connected to three existing variables. In the example shown, $c \equiv (u \vee v \vee \neg w)$ where c contains the pth occurrence of the boolean variable u in $I_{3\text{-SAT}}$, the qth occurrence of v and the rth occurrence of $\neg w$. In the gadget $H(c)$ shown in Fig. 6, $i(c)$ is connected to $i(\neg u, p)$, $i(\neg v, q)$ and $i(w, r)$.

Let $M_i = \max_{a \in d_i}(c'_i(a))$, where c' represents the constraint functions in V'. Each variable $i(c)$ can contribute at most 1 to $f_{\min}(V')$, i.e., $M_{i(c)} \leqslant 1$. By construction of

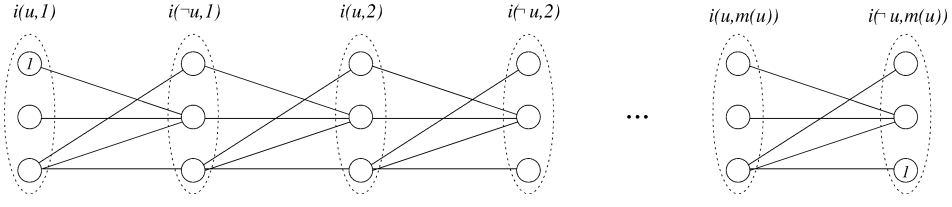

Fig. 5. The gadget $G(u)$ representing the m_u occurrences of u and the m_u occurrences of $\neg u$ in $I_{3\text{-SAT}}$.

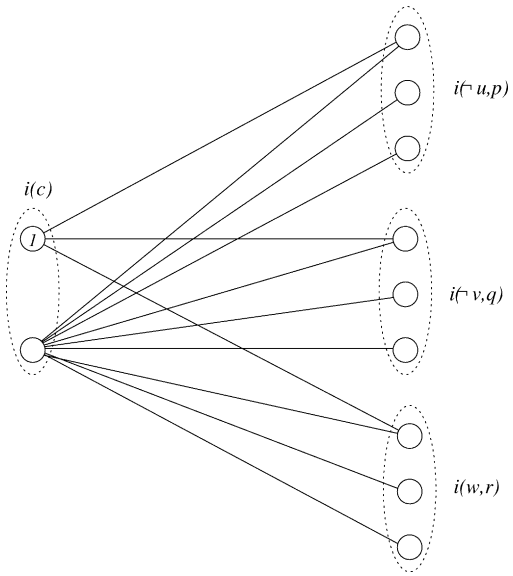

Fig. 6. The gadget $H(c)$ for $c \equiv (u \vee v \vee \neg w)$.

$G(u)$, if any variable (e.g., $i(u,r)$) contributes 1 to $f_{\min}(V')$, then its adjacent variables ($i(\neg u, r-1)$ and $i(\neg u, r)$) cannot contribute to $f_{\min}(V')$. This means that the maximum value of $f_{\min}(V')$ is $d + 3d/2 = 5d/2$, since the total number of variables in the gadgets $G(u)$ is $3d$. Indeed, $f_{\min}(V') = 5d/2$ iff for all clauses c, $M_{i(c)} = 1$ and for all boolean variables u,

$$\begin{aligned}&\big(\forall r \in \{1,\ldots,m(u)\},\ M_{i(u,r)} = 1 \wedge M_{i(\neg u,r)} = 0\big) \vee \\ &\big(\forall r \in \{1,\ldots,m(u)\},\ M_{i(\neg u,r)} = 1 \wedge M_{i(u,r)} = 0\big).\end{aligned} \tag{1}$$

Consider the clause $c \equiv (u \vee v \vee \neg w)$, whose gadget $H(c)$ is shown in Fig. 6. By construction of $H(c)$, $M_{i(c)} = 1$ implies that

$$(M_{i(\neg u,p)} = 0) \vee (M_{i(\neg v,q)} = 0) \vee (M_{i(w,r)} = 0)$$

which, in turn, implies from (1), that

$$(M_{i(u,1)} = 1) \vee (M_{i(v,1)} = 1) \vee (M_{i(w,1)} = 0). \tag{2}$$

Suppose that V has an arc consistency closure V' with $f_{\min}(V') \succcurlyeq 5d/2$. For each variable u in $I_{3\text{-SAT}}$, set $u = \mathit{true}$ iff $M_{i(u,1)} = 1$. For each clause c, for example $c \equiv (u \vee v \vee \neg w)$, we know from (2) that either $u = \mathit{true}$, $v = \mathit{true}$ or $w = \mathit{false}$. Hence $I_{3\text{-SAT}}$ is satisfied.

Suppose that ω is a model of $I_{3\text{-SAT}}$. In each gadget $G(u)$ in V and for each $r \in \{1,\ldots,m(u)\}$, if $u = \mathit{true}$ in ω then project penalties onto $i(u,r)$ from its constraints with the adjacent variables $i(\neg u, r-1)$ and $i(\neg u, r)$; if $u = \mathit{false}$ in ω, then project penalties onto $i(\neg u, r)$ from its constraints with the adjacent variables $i(u,r)$ and $i(u,r+1)$. Consider a gadget $H(c)$ in V, such as the gadget illustrated in Fig. 6 for $c \equiv (u \vee v \vee \neg w)$. If $u = \mathit{true}$ in ω, then project penalties onto $i(c)$ from its constraint with $i(\neg u, p)$; if

$u = \mathit{false}$ in ω, then project penalties onto $i(\neg u, p)$ from its constraint with $i(c)$. Similarly, if $\neg w = \mathit{true}$ in ω, then project penalties onto $i(c)$ from its constraint with $i(w, r)$; if $\neg w = \mathit{false}$ in ω, then project penalties onto $i(w, r)$ from its constraint with $i(c)$. Let V' be the resulting arc consistent VCSP. Since each clause is satisfied by ω, $M_{i(c)} = 1$. Furthermore, if $u = \mathit{true}$ in ω, then $M_{i(u,r)} = 1$ for $r \in \{1, \ldots, m(u)\}$ and if $u = \mathit{false}$ in ω, then $M_{i(\neg u,r)} = 1$ for $r \in \{1, \ldots, m(u)\}$. Thus $f_{\min}(V') = d + 3d/2 = 5d/2$. □

Even if the problem of finding a maximum AC-induced lower bound is NP-hard, it is nevertheless possible to use the lower bound induced by the arbitrary AC closure found by polynomial time AC enforcing. Although not optimal, such lower bounds have been found effective in the context of branch and bound algorithms for weighted MAX-CSPs and MAX-SAT problems in [6,14,17]. More work is needed to prove the existence of polynomial time approximation for this problem or even to find good heuristics to choose which equivalence-preserving operation to apply first during AC enforcing.

5.5. The case of strictly monotonic VCSPs

For strictly monotonic VCSPs, Algorithm 4 can be improved using an alternative equivalent definition of arc consistency based on the notion of the underlying CSP.

Definition 35. The *underlying CSP* of a VCSP V has the same variables and domains as V together with, for each constraint $c_P \in C$, a crisp constraint c'_P satisfying $\forall t \in \ell(P)$ $(t \in c'_P \Leftrightarrow c_P(t) \prec \top)$ (i.e., t is not a totally forbidden labelling).

In strictly monotonic VCSPs, the only idempotent elements are $\top$ and $\bot$. This allows us to give an equivalent but simpler definition of generalised arc consistency:

Theorem 36. *If $\oplus$ is a strictly monotonic operator, then a VCSP is generalised arc consistent iff*:

(1) *its underlying CSP is generalised arc consistent,*
(2) $\forall c_P \in C^+, \forall i \in P, \forall a \in d_i$, *if* $c_i(a) \prec \top$ *then* $\exists t \in \ell(P - \{i\})(c_P(t \cdot a) = \bot)$.

Proof. ($\Rightarrow$) Suppose that a strictly monotonic VCSP is generalised arc consistent but its underlying CSP is not. Then $\exists c_P \in C$, $i \in P$, $a \in d_i$ such that $(c_i(a) \prec \top) \wedge (\forall t \in \ell(P - \{i\})\ (c_P(t \cdot a) = \top \vee \exists j \in P - \{i\}\ (c_j(t_{\downarrow\{j\}}) = \top)))$. But, by condition (1) of Definition 26, $(c_j(t_{\downarrow\{j\}}) = \top)$ implies $c_P(t \cdot a) = \top$. Hence, $c_i(a) = \top$, by condition (2) of Definition 26, which is a contradiction. Suppose on the other hand that the VCSP is generalised arc consistent but condition (2) of Theorem 36 is not satisfied. Then $\exists c_P \in C^+$, $i \in P$, $a \in d_i$ such that $c_i(a) \prec \top$ and $\forall t \in \ell(P - \{i\})\ (c_P(t \cdot a) \succ \bot)$. But by condition (2) of Definition 26, for some $t \in \ell(P - \{i\})$, $c_i(a) = c_i(a) \oplus c_P(t \cdot a) \succ c_i(a)$ by strict monotonicity, which is impossible.

($\Leftarrow$) Suppose that a strictly monotonic VCSP satisfies conditions (1) and (2) of Theorem 36. Condition (2) clearly implies condition (2) of Definition 26. Suppose that condition (1) of Definition 26 is not satisfied. Then $\exists c_P \in C$, $t \in \ell(P)$ such that $c_P(t) \neq$

```
Enforce arc consistency in the underlying CSP
foreach c_P ∈ C^+ do
    foreach i ∈ P do
        foreach a ∈ d_i do Project(c_P, i, a)
```

Algorithm 5. Enforcing soft arc consistency on strictly monotonic VCSPs.

$\delta = (c_p(t) \oplus \beta) \ominus \beta$ where $\beta = \bigoplus_{j \in P} c_j(t_{\downarrow\{j\}})$. By Theorem 23, δ is idempotent, which is only possible if $c_j(t_{\downarrow\{j\}}) = \top$ for some $j \in P$. But the generalised arc consistency of the underlying CSP implies $c_P(t) = \top$, which provides the necessary contradiction. □

A possible way to enforce soft arc consistency on a strictly monotonic VCSP is therefore to first enforce classical arc consistency on the underlying CSP, assign a valuation of $\top$ to all deleted values in the original VCSP and then enforce the second property by applying Project(c_P, i, a) once for all $c_P \in C^+$, all $i \in P$ and all $a \in d_i$ (see Algorithm 5).

The only additional result needed to prove that this algorithm works is the following one:

Theorem 37. *Let V be a strictly monotonic VCSP whose underlying CSP is arc consistent. Then, $\forall c_P \in C^+, \forall i \in P, \forall a \in d_i$, the application of the equivalence-preserving transformation* Project(c_P, i, a) *yields a VCSP whose underlying CSP is unchanged* (*and therefore arc consistent*).

Proof. Project(c_P, i, a) cannot increase a valuation $c_i(a) \neq \top$ to $\top$ since arc consistency on the underlying CSP would have deleted (i, a) in this case and therefore we would have set $c_i(a) = \top$. It cannot decrease the valuation of any tuple such that $c_P(t \cdot a) = \top$ since $\forall \beta \in E, \top \ominus \beta = \top$. □

5.5.1. Improving space complexity

The space complexity of the arc consistency enforcing algorithm presented in the previous sections is always dominated by the space complexity of the modified constraints which requires $O(ed^r)$ valuations. This is extremely expensive, especially for arbitrary arity constraints.

It turns out however that this space requirement can be reduced to $O(edr)$ for strictly monotonic valuation structures. The idea is to leave the original constraints of the problem unmodified and to record the changes in an additional data structure.

Consider a constraint $c_P \in C$ and a tuple $t \in \ell(P)$. When enforcing AC, two types of operation may modify the valuation $c_p(t)$:

(1) for any variable $i \in P$ and for the value $a = t_{\downarrow\{i\}} \in d_i$, Project$(c_p, i, a)$ will decrease $c_p(t)$ to $c_P(t) \ominus \alpha$ where α is a specifically computed valuation;
(2) for any variable $i \in P$ and for the value $a = t_{\downarrow\{i\}} \in d_i$, Extend$(i, a, c_P)$ will increase $c_p(t)$ to $c_P(t) \oplus \beta$ where β is a specifically computed valuation.

If we do not want to explicitly store the modified valuations $c_P(t)$, we could instead use the original definition of c_P and for every variable $i \in P$, for every value $a = t_{\downarrow\{i\}}$, redo all the sequence of application of $\oplus$ and $\ominus$ with the corresponding arguments. Since there are at most r variables and d values, this would require storing rd sequences of operation/argument pairs in order to be able to recover the valuation of any tuple $t \in \ell(P)$. In the case of fair strictly monotonic valuation structures, an additional property will allow us to compress every sequence in just two valuations.

Theorem 38. *Let $S = (E, \oplus, \succcurlyeq)$ be a fair strictly monotonic valuation structure. For all $\alpha, \beta, \gamma \in S$ s.t. $\alpha \succcurlyeq \beta$, we have $(\alpha \ominus \beta) \oplus \gamma = (\alpha \oplus \gamma) \ominus \beta$.*

Proof. On one side, we have $[(\alpha \ominus \beta) \oplus \gamma] \oplus \beta = [\gamma \oplus (\alpha \ominus \beta)] \oplus \beta = (\gamma \oplus \alpha)$. Similarly, $[(\alpha \oplus \gamma) \ominus \beta] \oplus \beta = (\alpha \oplus \gamma)$. So, $[(\alpha \ominus \beta) \oplus \gamma] \oplus \beta = [(\alpha \oplus \gamma) \ominus \beta] \oplus \beta$. This implies that neither $[(\alpha \ominus \beta) \oplus \gamma] \oplus \beta \succ [(\alpha \oplus \gamma) \ominus \beta] \oplus \beta$ nor $[(\alpha \ominus \beta) \oplus \gamma] \oplus \beta \prec [(\alpha \oplus \gamma) \ominus \beta] \oplus \beta$. By strict monotonicity, this shows that either $(\alpha \ominus \beta) \oplus \gamma = (\alpha \oplus \gamma) \ominus \beta$ or $\beta = \top$. However, if $\beta = \top$ then $\alpha = \top$ and we still have $(\alpha \ominus \beta) \oplus \gamma = \top = (\alpha \oplus \gamma) \ominus \beta$. □

For each constraint c_P, for each variable $i \in P$, we will store two tables of d valuations noted Δ^+_{Pi} and Δ^-_{Pi}, initialised to $\bot$. Let $a \in d_i$:

- $\Delta^-_{Pi}[a]$ will contain the combination of all the valuations that have been projected up to now from c_P onto (i, a);
- $\Delta^+_{Pi}[a]$ will contain the aggregation of all the valuations that have been extended up to now from (i, a) to c_P.

It we denote by c^{or}_P the original definition of the constraint c_P, the valuation of a tuple $t \in \ell(P)$ in the modified constraint c_P can be obtained at any time by:

$$c_P(t) = c^{or}_P(t) \oplus \left(\bigoplus_{i \in P} \left(\Delta^+_{Pi}[t_{\downarrow\{i\}}] \ominus \Delta^-_{Pi}[t_{\downarrow\{i\}}] \right) \right).$$

Indeed, this is just a rearrangement of the sequence of operations that are performed by arc consistency enforcing: all application of $\oplus$ are brought to the left which is always possible thanks to Theorem 38 and because it can only increase the left member in each difference. Thanks to this, the space complexity is now reduced to $O(edr)$ instead of $O(ed^r)$ and our two basic equivalence-preserving transformations Project and Extend become space tractable even for large arity constraints defined using cost functions.

Algorithm 6 describes the procedures that implement these transformations with these data structures. They make the application of soft arc consistency enforcing to problems with large arity cost functions realistic, even if these cost functions are represented by analytical formulae. Regarding time complexity, Extend is reduced to $O(1)$ but since computing $c_P(t)$ requires $O(r)$ $\oplus$ operations, Project is $O(rd^{r-1})$ instead of $O(d^{r-1})$. This makes generalised arc consistency enforcing on strictly monotonic VCSPs $O(red^r)$ in time. For binary constraints, we recover the usual $O(ed^2)$ time and $O(ed)$ space complexities for arc consistency enforcing.

Procedure Project(c_{ij}, i, a)

$\alpha \leftarrow \min_{t \in \ell(P-\{i\})} (c_P^{or}(t \cdot a) \oplus \left(\bigoplus_{j \in P} (\Delta^+_{Pj}[(t \cdot a)_{\downarrow\{j\}}] \ominus \Delta^-_{Pj}[(t \cdot a)_{\downarrow\{j\}}]) \right)$

$c_i(a) \leftarrow c_i(a) \oplus \alpha$

$\Delta^-_{Pi}[a] \leftarrow \Delta^-_{Pi}[a] \oplus \alpha$

Procedure Extend(i, a, c_P)

$\alpha \leftarrow c_i(a)$

$\Delta^+_{Pi}[a] \leftarrow \Delta^+_{Pi}[a] \oplus \alpha$

$c_i(a) \leftarrow c_i(a) \ominus \alpha$

Algorithm 6. Projection and extension for soft AC enforcing.

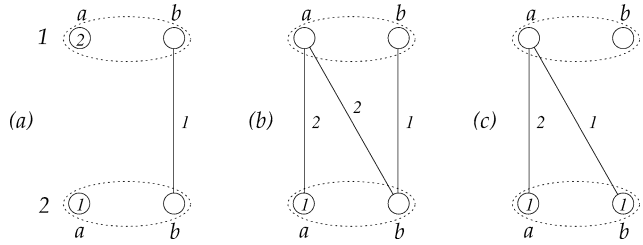

Fig. 7. Enforcing directional arc consistency.

6. Directional arc consistency

In CSPs, directional arc consistency is a weak version of arc consistency. In VCSPs, the order imposed on variables by directional arc consistency (first defined in [5] for strictly monotonic operators) makes it possible to use the unlimited version of Extend (instead of AC-Extend) and still get a terminating algorithm. For this reason, soft directional arc consistency may be stronger than arc consistency.

Consider the MAX-CSP in Fig. 7(a). It includes one binary constraint that forbids pair $((1, b)(2, b))$ and two unary constraints that forbid values $(1, a)$ and $(2, a)$. This VCSP is already arc consistent and the corresponding lower bound $f_{\min}(V)$ is equal to 0. However, we can apply Extend on value $(1, a)$, getting the equivalent VCSP in Fig. 7(b) which is not arc consistent. We can then apply Project on value $(2, b)$ and obtain the VCSP in Fig. 7(c) with a corresponding lower bound $f_{\min}(V) = 1$.

This improved lower bound has been obtained because we have decided to pool all the unary valuations on one of the variables. This can be done successively on all variables using any given variable order. In the context of branch and bound (or other tree-based search), weights can for example be propagated towards those variables which occur earlier in the instantiation order.

Definition 39. A binary VCSP is *directional arc consistent* according to an order $<$ on variables if $\forall c_{ij} \in C^+$ such that $i < j$, $\forall a \in d_i$

$$c_i(a) = \min_{b \in d_j} \big(c_i(a) \oplus c_{ij}(a, b) \oplus c_j(b)\big).$$

```
   Procedure DAC()
     for (i ← (n − 1) downto 1) do
       foreach (j ∈ X s.t. j > i and c_ij ∈ C) do
         {Start of propagation of c_ij}
         foreach (b ∈ d_j) do Extend(j, b, c_ij)
         foreach (a ∈ d_i) do Project(c_ij, i, a)
         {End of propagation of c_ij}
1        {Assert A(i, j): ∀a ∈ d_i, c_i(a) = min_{b∈d_j}(c_i(a) ⊕ c_ij(a, b) ⊕ c_j(b))}
```

Algorithm 7. Enforcing directional arc consistency on fair binary VCSPs.

Provided that the VCSP is fair, directional arc consistency can be established in polynomial time by the procedure DAC in Algorithm 7, where Project and Extend are as given in Algorithm 6.

Since DAC only applies equivalence-preserving transformations, it yields an equivalent VCSP. The following lemma is needed to prove that the VCSP obtained is directional arc consistent.

Lemma 40. *If $\oplus$ is fair, then*

$$(\alpha = \alpha \oplus \beta \oplus \gamma) \wedge (\alpha' \succcurlyeq \alpha) \wedge (\gamma' \preccurlyeq \gamma) \Rightarrow (\alpha' = \alpha' \oplus \beta \oplus \gamma').$$

Proof. Suppose that $(\alpha = \alpha \oplus \beta \oplus \gamma) \wedge (\alpha' \succcurlyeq \alpha) \wedge (\gamma' \preccurlyeq \gamma)$. Then $\alpha' \preccurlyeq \alpha' \oplus \beta \oplus \gamma' \preccurlyeq \alpha' \oplus \beta \oplus \gamma \preccurlyeq (\alpha' \ominus \alpha) \oplus (\alpha \oplus \beta \oplus \gamma) = (\alpha' \ominus \alpha) \oplus \alpha = \alpha'$. It follows that $\alpha' = \alpha' \oplus \beta \oplus \gamma'$. □

Theorem 41. *If the binary VCSP is fair, then directional arc consistency can be established in* $O(ed^2)$ *time and* $O(ed)$ *space complexity.*

Proof. The assertion $A(i, j)$ is clearly true during execution of DAC at line 1 of Algorithm 7 when constraint c_{ij} has just been propagated.

It suffices to show that $A(i, j)$ cannot be invalidated by later propagations of constraints $c_{i'j'}$ where $i' < j'$ and $(i' < i) \vee (i' = i \wedge j \neq j')$. Such operations may increase $c_i(a)$ and may decrease $c_j(b)$ but cannot modify $c_{ij}(a, b)$. From Lemma 40, $c_i(a) = c_i(a) \oplus c_{ij}(a, b) \oplus c_j(b)$ remains true and hence assertion $A(i, j)$ cannot be invalidated by later propagations and DAC yields a directional arc consistent VCSP.

As for time complexity, procedures Extend and Project are called $O(ed)$ times and are both $O(d)$ for binary VCSPs. DAC is therefore $O(ed^2)$ in time. The $O(ed)$ space complexity can be attained using the implementations of Project and Extend presented in Algorithm 6. □

When restricted to strictly monotonic VCSPs, Theorem 41 can be related to the result, proved in [5], that *full directional arc consistency*, a stronger version of directional arc consistency, can be established in $O(ed^2)$ time and space complexity. A VCSP is full directional arc consistent if and only if it is simultaneously arc consistent and directional arc consistent.

Theorem 42. *Suppose that the constraint graph of a binary fair VCSP V is a tree T and that V is directional arc consistent according to some topological ordering of the tree (i is the father of j in $T \Rightarrow (i < j)$). Then, for all $a \in d_1$, $c_1(a)$ is the optimal valuation over all solutions to V in which variable 1 is assigned value $a \in d_1$.*

Proof. Let S be the set of all the sons of variable 1 in T and for each $j \in S$, let T_j be the set of all variables in the subtree rooted in j. By induction, we assume that $\forall j \in S$, $\forall b \in d_j$, the valuation $c_j(b)$ is the optimal valuation over all solutions to the subproblem on T_j. Let t_{bj} be one corresponding optimal tuple over T_j.

Let $a \in d_1$ and for each $j \in S$, let b_{ja} be a value in d_j that minimises $c_1(a) \oplus c_{1j}(a, b_{ja}) \oplus c_j(b_{ja})$. Since $\forall i, j \in S$, $i \neq j$, $T_i \cap T_j = \emptyset$, we can build a tuple t over X by concatenation of each $t_{b_{ja}j}$ for all $j \in S$ and by assigning value a to variable 1. The valuation of the tuple t, is $\mathcal{V}_V(t) = c_1(a) \oplus \bigoplus_{j \in S}(c_j(b_{ja}) \oplus c_{1j}(a, b_{ja})) = c_i(a)$ since the VCSP is directional arc consistent.

Suppose there exists t' such that $t_{\downarrow\{1\}} = a$ and $\mathcal{V}_V(t') \prec \mathcal{V}_V(t)$. Since the problem is tree-structured, the valuation of t' can be written as

$$
\begin{aligned}
\mathcal{V}_V(t') &= c_1(a) \oplus \bigoplus_{j \in S}\left(c_{1j}(a, t'_{\downarrow\{j\}}) \oplus \mathcal{V}_{V(T_j)}(t'_{\downarrow T_j})\right) \quad \text{(tree structure)} \\
&\succcurlyeq c_1(a) \oplus \bigoplus_{j \in S}\left(c_{1j}(a, t'_{\downarrow\{j\}}) \oplus c_j(t'_{\downarrow\{j\}})\right) \quad \text{(induction, monotonicity)} \\
&\succcurlyeq c_1(a) \oplus \bigoplus_{j \in S}\left(c_j(b_{ja}) \oplus c_{1j}(a, b_{ja})\right) \quad \text{(definition of } b_{ja}\text{)} \\
&= c_1(a) = \mathcal{V}_V(t)
\end{aligned}
$$

which shows that no such t' exists. □

Enforcing directional arc consistency is actually strongly related to the computation of so-called mini-buckets [8]. On binary MAX-CSP, it can easily be shown that the lower-bound induced by mini-buckets involving at most 2 variables is the same as the lower bound induced by directional arc consistency. It is however more interesting to enforce directional arc consistency since this will provide both a lower bound and a directional arc consistent equivalent problem. All the work done to compute the lower bound is captured in this directionally arc consistent closure which offers the opportunity to perform incremental updates of the lower bound if new constraints are added to the problem. This is exactly the case in the context of branch and bound search [6,17].

7. Arc irreducibility

We have seen that the cost of generalising arc consistency from crisp to valued constraint satisfaction problems is the loss of uniqueness of the arc consistent closure. As Fig. 4 showed, two different arc consistent closures may also induce different lower bounds via $f_{\min}$.

If a VCSP V is equivalent to another VCSP V' which is better than V (according to some formally defined criterion $\mathcal{C}$) then we say that V is reducible for this criterion. Reducing a VCSP to an equivalent irreducible problem is, in general, an NP-hard problem. To see this, consider a CSP. If it is inconsistent (i.e., has no solution), then its best equivalent problem, for any reasonable criterion $\mathcal{C}$, is a CSP in which this fact is made explicit, for example, by having $\forall a \in d_i$, $c_i(a) = \top$ for some variable $i \in X$. Since testing consistency of a CSP is an NP-complete problem, we can deduce that testing global irreducibility is NP-hard.

Fortunately, irreducibility has local versions which are analogous to local consistency. For example, a binary VCSP V is arc-irreducible if, for all pairs of variables $i, j \in X$, V cannot be improved by replacing c_i, c_j, c_{ij} by an equivalent set of constraints c'_i, c'_j, c'_{ij}. However, before giving a formal definition of arc-irreducibility, we have to consider which criteria we could use to compare equivalent VCSPs.

Definition 43. A *problem evaluation function* f is a function which, for each VCSP V, assigns a value to V in a totally ordered range. When comparing two equivalent VCSPs, V_1 and V_2, V_1 is considered as a better expression of the problem if $f(V_1) > f(V_2)$.

Example 44. The function $f_{\min}$ previously defined as

$$f_{\min}(V) = \bigoplus_{c_P \in C} \Big(\min_{t \in \ell(P)i} \big(c_P(t)\big)\Big)$$

is a problem evaluation function.

The main result presented in this section concerns $f_{\min}$, but the definitions are valid for any problem evaluation function f.

Definition 45. A binary VCSP V is *arc-irreducible* with respect to the problem evaluation function f (or $(2, f)$-irreducible), if $\forall J \subseteq X$, $|J| = 2$, for any VCSP V' derived from V by an equivalence-preserving transformation on J, $f(V) \geqslant f(V')$.

Note that for certain choices of the problem evaluation function f, an arc-irreducible VCSP is not necessarily arc-consistent. For example, if $\forall i, j \in X$, $\exists a \in d_i$, $b \in d_j$ such that $c_i(a) = c_j(b) = c_{ij}(a, b) = \bot$, then the VCSP is $(2, f_{\min})$-irreducible but it is not necessarily arc-consistent. Conversely, the VCSP in Fig. 4(c) is arc consistent, but not $(2, f_{\min})$-irreducible. The following theorem shows that there is an important relationship between directional arc consistency and $(2, f_{\min})$-irreducibility.

Theorem 46. *A fair binary VCSP $V = \langle X, D, C, S\rangle$ which is directional arc consistent is $(2, f_{\min})$-irreducible.*

Proof. Suppose that V is directional arc consistent and let $i, j \in X$ be such that $i < j$. Then, by Definition, $\forall a \in d_i$, $\exists b \in d_j$ such that

$$c_i(a) = \min_{b \in d_j} \big(c_i(a) \oplus c_{ij}(a, b) \oplus c_j(b)\big).$$

Thus the minimum valuation of a solution to the subproblem of V on $\{i, j\}$ is

$$\min_{a \in d_i} \big(c_i(a)\big).$$

Suppose that this minimum is attained for $a = u \in d_i$ and that $v \in d_j$ is such that

$$c_i(u) = \big(c_i(u) \oplus c_{ij}(u, v) \oplus c_j(v)\big). \tag{3}$$

Now consider a VCSP V' obtained by an equivalence-preserving transformation of V on $\{i, j\}$ which replaces c_i, c_j, c_{ij} by c'_i, c'_j, c'_{ij}. Then

$$f_{\min}(V') = \left(f_{\min}(V) \ominus \left(\bigoplus_{P \subseteq \{i,j\}} \big[\min_{t \in \ell(P)} c_P(t) \big] \right) \right) \oplus \left(\bigoplus_{P \subseteq \{i,j\}} \big[\min_{t \in \ell(P)} c'_P(t) \big] \right)$$

since V and V' only differ on $\{i, j\}$. Thus

$$\begin{aligned} f_{\min}(V') &\preccurlyeq \big(f_{\min}(V) \ominus c_i(u)\big) \oplus \big(c'_i(u) \oplus c'_{ij}(u, v) \oplus c'_j(v)\big) \\ &\preccurlyeq \big(f_{\min}(V) \ominus \big(c_i(u) \oplus c_{ij}(u, v) \oplus c_j(v)\big)\big) \oplus \big(c'_i(u) \oplus c'_{ij}(u, v) \oplus c'_j(v)\big) \end{aligned}$$

by Eq. (3). Thus $f_{\min}(V') \preccurlyeq f_{\min}(V)$ due to the equivalence of the subproblems of V and V' on $\{i, j\}$. Thus, V is $(2, f_{\min})$-irreducible. □

This result must be considered with care. Given any VCSP V, Theorem 46 states that any directional arc consistent closure V_{dac} of V is always $(2, f_{\min})$-irreducible. However, an arc consistent closure V_{ac} may exist such that $f_{\min}(V_{ac}) \succ f_{\min}(V_{dac})$.

Corollary 47. *Arc-irreducibility with respect to $f_{\min}$ can be established in $O(ed^2)$ time complexity and $O(ed)$ space complexity on fair binary VCSPs.*

Proof. This follows directly from Theorems 41 and 46. □

8. Conclusions

The concept of arc consistency plays an essential role in constraint satisfaction as a problem simplification operation and as a tree-pruning technique during search through the detection of local inconsistencies among the uninstantiated variables. We have shown that it is possible to generalise arc consistency to any instance of the valued CSP framework provided the operator for aggregating penalties allows to compute differences between penalties.

A polynomial-time algorithm for establishing soft arc consistency exists. Its space and time complexity is identical to that of establishing arc consistency in CSPs whenever the aggregation operator of the VCSP is strictly monotonic, which is the case in MAX-CSP, for example. Contrarily to classical CSP arc consistency, it does not define a unique arc consistency closure. This algorithm nevertheless provides an efficient technique for generating lower bounds on the value of a solution which can be used during branch-and-bound search as in [15,16]. The problem of finding the maximum lower bound is NP-hard

but using arbitrary (not necessarily maximum) lower bounds has been found effective to solve both MAX-CSP [14,17] and MAX-SAT problem [6].

We have also defined a directional version of soft arc consistency which is potentially stronger since it allows non-local propagation of penalties. Directional soft arc consistency implies a form of local optimality in the expression of the VCSP, called arc irreducibility. Furthermore, the complexity of establishing directional arc consistency is identical to that of establishing arc consistency in CSPs.

Acknowledgement

We thank Javier Larrosa and the anonymous referees for their comments on the paper.

References

[1] M.S. Affane, H. Bennaceur, A weighted arc consistency technique for Max-CSP, in: Proc. 13th ECAI, Brighton, UK, 1998, pp. 209–213.
[2] S. Bistarelli, R. Gennari, F. Rossi, General properties and termination conditions for soft constraint propagation, Constraints 8 (1) (2003) 79–97.
[3] S. Bistarelli, H. Fargier, U. Montanari, F. Rossi, T. Schiex, G. Verfaillie, Semiring-based CSPs and valued CSPs: Frameworks, properties and comparison, Constraints 4 (1999) 199–240.
[4] S. Bistarelli, U. Montanari, F. Rossi, Constraint solving over semirings, in: Proc. 14th IJCAI, Montreal, QB, 1995, pp. 624–630.
[5] M.C. Cooper, Reduction operations in fuzzy or valued constraint satisfaction, Fuzzy Sets and Systems 134 (3) (2003) 311–342.
[6] S. de Givry, J. Larrosa, P. Meseguer, T. Schiex, Solving Max-Sat as weighted CSP, in: Proc. Ninth International Conference on Principles and Practice of Constraint Programming, Kinsale, Ireland, in: Lecture Notes in Computer Science, Springer, Berlin, 2003.
[7] S. de Givry, G. Verfaillie, T. Schiex, Bounding the optimum of constraint optimization problems, in: Proc. Third International Conference on Principles and Practice of Constraint Programming, Schloss Hagenberg, Austria, 1997, pp. 405–419.
[8] R. Dechter, Mini-buckets: A general scheme for generating approximations in automated reasoning, in: Proc. IJCAI-97, Nagoya, Japan, 1997, pp. 1297–1303.
[9] K. Evans, M. Konikoff, R. Mathis, J. Maden, G. Whipple, Totally ordered commutative monoids, Semigroup Forum 62 (2) (2001) 249–278.
[10] H. Fargier, J. Lang, T. Schiex, Selecting preferred solutions in Fuzzy Constraint Satisfaction Problems, in: Proc. 1st European Congress on Fuzzy and Intelligent Technologies, 1993.
[11] E. Freuder, R. Wallace, Partial constraint satisfaction, Artificial Intelligence 58 (1992) 21–70.
[12] E. Klement, R. Mesiar, E. Pap, Triangular Norms, Kluwer Academic, Dordrecht, 2000.
[13] A.M. Koster, Frequency assignment: Models and algorithms, PhD Thesis, University of Maastricht, The Netherlands, available at http://www.zib.de/koster/thesis.html, November 1999.
[14] J. Larrosa, On arc and node consistency in weighted CSP, in: Proc. AAAI-02, Edmondton, AB, 2002, pp. 48–53.
[15] J. Larrosa, P. Meseguer, T. Schiex, Maintaining reversible DAC for Max-CSP, Artificial Intelligence 107 (1) (1999) 149–163.
[16] J. Larrosa, P. Meseguer, T. Schiex, G. Verfaillie, Reversible DAC and other improvements for solving Max-CSP, in: Proc. AAAI-98, Madison, WI, 1998, pp. 347–352.
[17] J. Larrosa, T. Schiex, In the quest of the best form of local consistency for weighted CSP, in: Proc. IJCAI-03, Acapulco, Mexico, 2003, pp. 239–244.

[18] R. Mohr, G. Masini, Good old discrete relaxation, in: Proc. ECAI-88, Munchen, Germany, 1988, pp. 651–656.
[19] A. Rosenfeld, R. Hummel, S. Zucker, Scene labeling by relaxation operations, IEEE Trans. Systems Man Cybernet. 6 (6) (1976) 173–184.
[20] T. Schiex, Maximizing the reversible DAC lower bound in Max-CSP is NP-hard, Techical Report 1998/02, INRA, July 1998.
[21] T. Schiex, Possibilistic constraint satisfaction problems or "How to handle soft constraints?", in: Proc. 8th Internat. Conf. on Uncertainty in Artificial Intelligence, Stanford, CA, 1992, pp. 268–275.
[22] T. Schiex, Arc consistency for soft constraints, in: Principles and Practice of Constraint Programming—CP 2000, Singapore, in: Lecture Notes in Computer Science, vol. 1894, Springer, Berlin, 2000, pp. 411–424.
[23] T. Schiex, H. Fargier, G. Verfaillie, Valued constraint satisfaction problems: Hard and easy problems, in: Proc. 14th IJCAI, Montreal, QB, 1995, pp. 631–637.
[24] L. Shapiro, R. Haralick, Structural descriptions and inexact matching, IEEE Trans. Pattern Anal. Machine Intelligence 3 (1981) 504–519.
[25] R. Wallace, Directed arc consistency preprocessing, in: M. Meyer (Ed.), Selected papers from the ECAI-94 Workshop on Constraint Processing, in: Lecture Notes in Computer Science, vol. 923, Springer, Berlin, 1995, pp. 121–137.