Domain Reduction for Valued Constraints by Generalising Methods from CSP

Martin Cooper, Wafa Jguirim, David Cohen

Domain Reduction for Valued Constraints by Generalising Methods from CSP

Martin C. Cooper[1(✉)], Wafa Jguirim[1,2], and David A. Cohen[3]

[1] IRIT, University of Toulouse III, Toulouse, France
{cooper,Wafa.Jguirim}@irit.fr
[2] National School of Computer Science, University of Manouba, Manouba, Tunisia
[3] Department of Computer Science, Royal Holloway, University of London, Egham, UK
d.cohen@rhul.ac.uk

Abstract. For classical CSPs, the absence of broken triangles on a pair of values allows the merging of these values without changing the satisfiability of the instance, giving experimentally verified reduction in search time. We generalise the notion of broken triangles to VCSPs to obtain a tractable value-merging rule which preserves the cost of a solution.

We then strengthen this value merging rule by using soft arc consistency to remove soft broken triangles and we show that the combined rule generalises known notions of domain value substitutability and interchangeability. Unfortunately the combined rules are no longer tractable to apply, but can still have applications as heuristics for reducing the search space.

Finally we consider the generalisation of another value-elimination rule for CSPs to binary VCSPs. This new rule properly generalises neighbourhood substitutability and so we expect it will also have practical applications.

Keywords: Valued Constraint Satisfaction Problem · Value merging Value elimination · Tractability

1 Introduction, Notation and Definitions

Constraint Satisfaction (CSP) has had a significant impact on our ability to solve large and practical declarative problems, for example crew scheduling [14], and online workflow allocation [20]. However, in such complex problem domains, it has been restricted in its applicability by not being able to express degrees of satisfaction. The natural extension, valued constraint satisfaction (VCSP), allows us to express preferences amongst assignments and so forms a useful paradigm, extending the CSP to optimisation, whilst maintaining the declarative feel, allowing us to reason directly in problem domains.

Partially supported by ANR-11-LABX-0040-CIMI within the French *Agence Nationale de la Recherche* program ANR-11-IDEX-0002-02.

https://doi.org/10.1007/978-3-319-98334-9_5

Since these problems are NP-hard much effort has been applied to find algorithms and techniques to reduce the search space. Such techniques are most effective if they do not change the set of possible solutions, but simply avoid exploring search avenues that we know cannot be productive. Key amongst these techniques are propagation and symmetry reduction, which can both be applied either before or during search. Search-space reduction has been approached in two ways: by the application of group theory [1,3,11,19,21] to identify equivalent branches of the search tree, or by using local structure to prune values or variables from the search [2,6,7,9,16,18]. It is the latter approach that we continue to develop in this paper, extending domain reduction results from constraint satisfaction to valued constraint satisfaction [5,17]. Indeed we will be combining domain reduction methods from consistency approaches [4,8] which have been essential in making global constraints effective [15] with local pattern based value merging to make our techniques more widely applicable.

We observe that there are often several ways to extend reduction techniques from CSP to VCSP. This is most notably the case for arc consistency, which has been generalised to distinct techniques, FDAC, EDAC and VAC, which all coincide with CSP arc consistency when applied directly to the classical CSP, but correspond to distinct levels of soft consistency on the VCSP [8].

Similarly, different generalisations of neighbourhood substitution have been proposed for VCSPs [5,13,17], including a stronger condition for substitutability taking into account the current lower and upper bounds [12]. In this paper we consider generalisations from CSPs to VCSPs of value-elimination rules based on the merging of values [7,18].

1.1 Definitions

A VCSP instance is a collection of cost functions applied to sets of variables [8].

For simplicity of presentation, in this paper we assume that costs are taken from the non-negative rationals together with the special cost infinity (∞). Of course, such instances are precisely equivalent to classical (crisp) CSP instances when the costs happen to all be either 0 or ∞. For simplicity of notation we always assume that the set of variables of the instance I is $V(I) = \{X_1^I, \ldots, X_n^I\}$, allowing us to use indexes to refer to variables. The domain of possible values for the variable X_i^I is denoted by D_i^I.

A subset of the variables is called a scope. An assignment to scope σ maps each $X_i^I \in \sigma$ to an element of its domain, D_i^I. The instance I includes a set of cost functions $C(I) = \{\phi_\sigma^I \mid \sigma \in S(I)\}$ where $S(I)$ is a set of scopes and each $\phi_\sigma^I \in C(I)$ maps assignments (to the variables σ) to costs. When we refer to a cost function ϕ_σ^I for which $\sigma \notin S(I)$ we always mean the cost function that is identically zero.

The cost of an assignment s to a set of variables $Y \subseteq V(I)$ is given by

$$cost_Y^I(s) \;=\; \sum_{\sigma \subseteq Y} \phi_\sigma^I(s|_\sigma).$$

A solution is an assignment to $V(I)$ that minimizes cost.

Where it improves readability we omit the name of the instance when referring to domains and costs. We simplify notation by using ϕ_i, ϕ_{ij} to denote $\phi_{\{X_i\}}$ and $\phi_{\{X_i,X_j\}}$ respectively. Furthermore, for single variables X_i we use the assignment on $\{X_i\}$ mapping X_i to a interchangeably with the value a, since the meaning is always clear from the context.

Analogously to consistency propagation in the CSP, any soft consistency operation on a VCSP instance [8] alters the cost functions while preserving all solutions. A soft arc consistency (SAC) operation at variable X_i replaces the unary cost ϕ_i with ϕ'_i different only at domain value $d \in D_i$ where $\phi'_i(d) = \phi_i(d)+\alpha$, ($\alpha$ may be negative). To compensate, it replaces one other cost function ϕ_σ for which $X_i \in \sigma$ with ϕ'_σ where $\phi'_\sigma(s) = \phi_\sigma(s)$ except when $s(X_i) = d$ in which case $\phi'_\sigma(s) = \phi_\sigma(s) - \alpha$. In order to be well defined the result of this SAC operation must leave all costs non-negative: we are not allowed to subtract a larger cost from any existing cost. To make this definition subsume (crisp) arc-consistency we extend the definition of subtraction so that $\infty - \infty = \infty$.

This is illustrated in Fig. 1, where as usual, variables are (grey) ellipses containing domain value: cost pairs and non-zero (binary) costs are shown with labelled arcs. A solution to this instance assigns c to X_1, e to X_2 and f to X_3 and has cost 4.

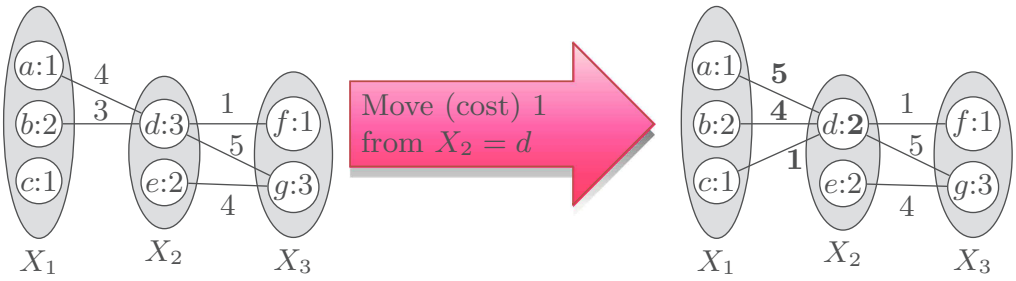

Fig. 1. SAC: moving cost 1 from $X_2 = d$ to $\phi_{1,2}$. Changed costs are highlighted.

Definition 1. *In a VCSP instance, a* GASBT (general-arity soft broken triangle) *on values a, b for X_i is an assignment s to the union of distinct non-empty scopes σ and ρ, where $X_i \notin \sigma \cup \rho$, such that*

$$\phi_{\sigma\cup\{X_i\}}(s|_\sigma \cup a) < \phi_{\rho\cup\{X_i\}}(s|_\rho \cup b)$$
$$\phi_{\rho\cup\{X_i\}}(s|_\rho \cup a) > \phi_{\sigma\cup\{X_i\}}(s|_\sigma \cup b)$$
$$cost_{\sigma\cup\rho}(s) < \infty$$

For binary CSP instances (where the costs lie in $\{0, \infty\}$), the notion of a general arity soft broken triangle coincides precisely with the classical CSP notion of broken triangle [7]. This correspondence is shown graphically in Fig. 2. In this figure the crisp CSP instance has dashed arcs to indicate disallowed tuples (cost $= \infty$) and solid arcs to indicate permitted tuples (cost $= 0$). Hence when σ and ρ each contain just one variable we say that a GASBT is a soft broken triangle (SBT).

Although the definition of a GASBT is independent of the unary costs $\phi_i(a), \phi_i(b)$, these costs will be critical in the definition of a value-merging rule.

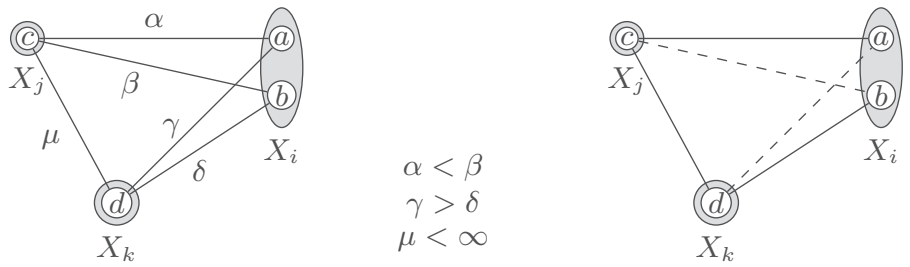

Fig. 2. A soft broken triangle and the corresponding crisp BTP pattern.

Definition 2. *The VCSP instance J obtained from I by* merging *a value pair $a, b \in D_i$ to produce a new value c has the same variables and domains as I except that $D_i^J = \big(D_i^I - \{a, b\}\big) \cup \{c\}$.*

The cost functions in J are defined as follows:

$$\phi_\sigma^J(t) \;= \begin{cases} \phi_\sigma^I(t) & \text{If } X_i \notin \sigma, \text{ or } t(X_i) \neq c \\ \min\{\phi_\sigma^I(t \cup a), \phi_\sigma^I(t \cup b)\} & \text{Otherwise.} \end{cases}$$

2 Value-Merging Rules

We say that $a, b \in D_i$ are *mergeable* in a VCSP instance if the cost of a solution to the new instance (after the merging of a and b) is identical to the cost of a solution to the original instance. We have the following rule.

Proposition 1. *Whenever $\phi_i(a) = \phi_i(b)$ and there is no general arity soft broken triangle on a, b, then $a, b \in D_i$ are mergeable. Furthermore, given a solution to the instance resulting from the merging of two values, we can find a solution to the original instance in linear time.*

Proof. Suppose that there is no GASBT on $a, b \in D_i$ in I, and let J be identical to I except that a, b have been merged to produce the value c. If a solution to J does not have value c at X_i^J then it is also a solution to I and we have nothing to prove. So, let s be a solution to J which assigns c to X_i^J. Denote by s_a, s_b the assignments in I which are identical to s except that X_i^I is assigned a or b (respectively).

It is clear from the definition of the cost functions in the merged instance that the cost of s in J is at most the minimum of the costs of s_a and s_b in I. If the cost of s is infinite, we have nothing to prove, so we assume it is finite.

Since there is no GASBT on a, b, and this is true for every pair of scopes σ and ρ, we must have either

$$\forall \sigma, \rho \text{ where } X_i^I \notin \sigma \cup \rho, \phi^I_{\sigma \cup \{X_i^I\}}(s|_\sigma \cup a) \leq \phi^I_{\rho \cup \{X_i^I\}}(s|_\rho \cup b)$$

or

$$\forall \sigma, \rho \text{ where } X_i^I \notin \sigma \cup \rho, \phi^I_{\rho \cup \{X_i^I\}}(s|_\rho \cup a) \geq \phi^I_{\sigma \cup \{X_i^I\}}(s|_\sigma \cup b)$$

Without loss of generality, suppose it is the former. Then

$$\forall \sigma, \text{where } X_i^J \notin \sigma, \quad \phi^J_{\sigma \cup \{X_i^J\}}(s|_\sigma \cup c) \;=\; \phi^I_{\sigma \cup \{X_i\}}(s|_\sigma \cup a)$$

Now, since $\phi_i^I(a) = \phi_i^I(b)$ we can replace c by a in s to obtain a solution to the original instance with the same global cost as s.

So, reconstructing a solution to I simply requires checking which of s_a or s_b is a solution to I. This can be achieved in time which is linear in the size of I.

We use the term SBT-merging (Soft Broken Triangle merging) for the merging of two values in a binary VCSP instance, allowed by the premise of Proposition 1.

2.1 Applying GASBTP Value Merging

For any $k \geq 1$, applying k-consistency operations until convergence to a CSP instance produces a unique closure [4]. Similarly, applying neighbourhood substitution operations [9] until convergence to a CSP instance produces a unique closure modulo isomorphism [5]. For VCSPs, finding the closure by soft arc consistency operation is not unique, but the problem of finding the best closure can be solved in polynomial time by linear programming [8]. It is therefore natural to ask the question of the uniqueness of and the complexity of finding the best closure of a VCSP instance under SBT merging operations. It turns out that the answer depends on whether the VCSP instance has infinite costs or not.

Theorem 1. *For a finite-valued VCSP (i.e. with no infinite costs), closure under GASBT-merging is unique up to value renaming. This closure can be found in polynomial time if the scopes are of bounded size. For general-valued VCSPs, maximizing the number of SBT-merges is NP-hard.*

Proof. For finite-valued VCSPs, it suffices to notice that GASBT-merging is equivalent to eliminating values $a \in D_i$ if $\exists b \in D_i$ such that $\forall \sigma, X_i \notin \sigma$ and for all assignments t to $\sigma \cup \{X_i\}$ we have that $\phi_{\sigma \cup \{X_i\}}(t \cup a) \geq \phi_{\sigma \cup \{X_i\}}(t \cup b)$. Clearly, elimination of such a value a cannot prevent eliminations at the same or other variables by the same rule. Of course we are free to name new domain values in any way that we choose, so we cannot guarantee that value merging ends up with precisely the same VCSP instance. However, it does not matter in which order we apply GASBT value merges to a finite-valued VCSP instance: we always converge to isomorphic instances.

Testing whether there is a GASBT on a pair of values at X_i for a given pair of scopes σ and ρ requires testing that a VCSP on $\sigma \cup \rho$ has finite value. In a finite valued VCSP this test is trivial. If either $\sigma \cup \{X_i\}$ or $\rho \cup \{X_i\}$ contains more variables than the bound on the size of a scope then the associated cost function is identically zero, so cannot satisfy the conditions required for a GASBT.

So, the existence of a GASBT value merge can be tested in polynomial time if we bound the arity of cost functions. Hence, the closure is unique modulo isomorphism and can be found in polynomial time by a greedy algorithm.

To show NP-hardness of optimal value merging when infinite costs are allowed observe that crisp binary CSP instances are precisely those binary VCSP instances with costs restricted to $\{0, \infty\}$. Since an SBT in such a binary VCSP is precisely a crisp broken triangle, SBT value merging is simply BT-merging in the corresponding CSP instance. It is known that finding the maximum number of BT-merges in a CSP instance is NP-hard (even for domains of size 3) [7]. It follows that finding the maximum number of SBT-merges in VCSPs is NP-hard, even if all costs belong to $\{0, \infty\}$, domains are of size 3, and scopes are binary.

3 Combining Soft Arc Consistency and SBT-Merging

GASBT value merging can only be performed when we have two domain values with identical unary costs. However, SAC operations allow us to move costs away from domain values. It is therefore possible that we can merge values, but only after performing the correct sequence of SAC operations. We will show two practical examples (with low complexity) where this does indeed occur and then show that, in general it is NP-hard to find such a sequence of SAC operations. In fact we will show, in the first result, that SAC followed by value merging subsumes a (natural but weak) form of valued neighbourhood substitution [5,17].

Definition 3. *We say that a is weak neighbourhood substitutable for b at variable X_i if every cost associated with a tuple assigning value b to X_i is not made worse by substituting value a. That is,*

For all σ, $X_i \notin \sigma$ for all assignments t to σ we have

$$\phi_{\sigma \cup \{X_i\}}(t \cup a) \leq \phi_{\sigma \cup \{X_i\}}(t \cup b).$$

Example 1. Consider again the binary VCSP instance in Fig. 1. Before the SAC operation the two values for variable X_2 cannot be GASBT value merged as they have different unary costs. On the other hand, even before the SAC operation, e is weak neighbourhood substitutable for d.

Since there are no infinite costs in this VCSP, testing for value merging here amounts to checking that the costs associated with value d are always at least as high as those associated with value e.

Now consider the binary VCSP instance shown in Fig. 3. Since the value $\phi_{12}(a, c) < \phi_{12}(b, c)$ but $\phi_{12}(a, e) > \phi_{12}(b, e)$ neither a nor b can be weak neighbourhood substituted for the other. On the other hand a and b can be GASBT value merged since $\phi_{23}(c, e) = \infty$.

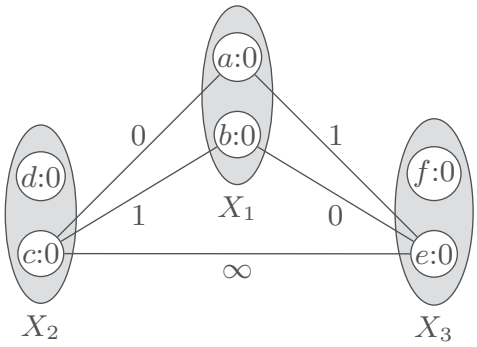

Fig. 3. In this binary VCSP instance a and b can be GASBT value merged, but they are not weak neighbourhood substitutable.

Notice that after the SAC operation the two values e and d in Fig. 1 can be GASBT merged. We now prove that this holds in general: weak substitutable values can always be GASBT merged after precisely one SAC operation.

Theorem 2. *If a is weak neighbourhood substitutable for b then, in polynomial time, we can find a SAC operation after which a and b can be GASBT value merged.*

Proof. Suppose that a is weak neighbourhood substitutable for b and that $\phi_i(a) < \phi_i(b)$. Choose any scope ρ and use a SAC operation to move the cost $\alpha = \phi_i(b) - \phi_i(a)$ from b to ϕ_ρ, where $X_i \in \rho$, replacing ϕ_i with ϕ'_i and ϕ_ρ with ϕ'_ρ and leaving all other cost functions unchanged.

Before the SAC operation we have, for all σ, $X_i \notin \sigma$ and for all assignments t to σ:

$$\phi_{\sigma\cup\{X_i\}}(t \cup a) \leq \phi_{\sigma\cup\{X_i\}}(t \cup b).$$

So, after the SAC operation, for all assignments t to $\rho - \{X_i\}$:

$$\begin{aligned}
\phi'_\rho(t \cup b) &= \phi_\rho(t \cup b) + \alpha && \textit{the SAC operation} \\
&> \phi_\sigma(t \cup b) && \textit{since } \phi_i(a) < \phi_i(b) \\
&\geq \phi_\sigma(t \cup a) && \textit{since } a \textit{ is substitutable for } b \\
&= \phi'_\sigma(t \cup a).
\end{aligned}$$

So the SAC operation preserves the weak neighbourhood substitutability, but now both a and b have equal unary cost at X_i.

We complete the proof by showing that if a is weak neighbourhood substitutable for b then there cannot be any GASBT on a and b. In fact this is trivially true since it can never occur for any scope ρ and assignment s that:

$$\phi_{\rho\cup\{X_i\}}(s|_\rho \cup a) > \phi_{\sigma\cup\{X_i\}}(s|_\sigma \cup b)$$

Since the SAC operation was entirely determined by the unary costs of a and b and σ was arbitrary this is polynomial time.

In fact we can identify another simple case in which we can immediately find a soft arc consistency operation which leads to GASBT-merging.

Definition 4. *We say that* $a, b \in D_i$ *are* almost interchangeable *if there is a scope* σ *with* $X_i \notin \sigma$ *such that for all scopes* $\rho \neq \sigma$ *with* $X_i \notin \rho$ *we have, for every assignment* t *to* ρ, $\phi_\rho(t \cup a) = \phi_\rho(t \cup b)$.

This definition of almost interchangeability is independent of the unary cost functions but still allows GASBT value merging after an appropriate SAC operation.

Proposition 2. *If* $a, b \in D_i$ *are almost interchangeable, then, in polynomial time, we can find a SAC operation after which* a *and* b *can be GASBT merged.*

Proof. We can simply make $\phi_i(a) = \phi_i(b)$ using a SAC operation which sends the difference in their cost to the cost function ϕ_σ. This leaves a and b almost interchangeable.

Since there is only one scope for which the cost functions can differ when a is replaced by b at X_i there can be no GASBT on a and b and they can now be merged.

When the costs lie in $\{0, \infty\}$, the notion of almost interchangeability coincides with the notion of virtual interchangeability [18].

Corollary 1. *In a binary VCSP, suppose there is a variable* X_j $(j \neq i)$ *such that* $\forall k \notin \{i, j\}$, $\forall c \in D_k$, $\phi_{ik}(a, c) = \phi_{ik}(b, c)$ *then* a *and* b *are* almost interchangeable *and can be merged.*

Having shown the usefulness of applying SAC operations to remove any occurrences of GASBT we conclude the section with a proof that it is in general NP-hard to determine whether such SAC operations can be found.

Suppose that we have a binary VCSP instance I with a unary cost function ϕ_i for which $\phi_i(a) > \phi_i(b)$. Our problem is to find a set of costs $\{q_j \mid j \neq i\}$, one cost for each (other) variable X_j, such that by sending each cost q_j from $\phi_i(a)$ to the cost function ϕ_{ij}, we obtain an instance with $\phi_i(a) = \phi_i(b)$ that has no soft broken triangle on a and b at X_i.

We now prove that this problem is NP-hard.

Theorem 3. *Given a VCSP instance* I *and variable* X *with domain values* a *and* b, *it is NP-hard to determine whether there are soft arc consistency operations on* a *and* b *which make the unary costs of* a *and* b *equal whilst also eliminating all SBT occurrences on* a *and* b.

Proof. We provide a polynomial reduction from the problem SubsetSum which is well known to be NP-complete [1,10]. An instance $\langle S, M \rangle$ of SubsetSum consists of a set S of positive integers and an integer M. The corresponding question is whether there exists a subset $T \subseteq S$ whose elements sum to M, i.e.

$$\sum_{s \in T} s \;=\; M.$$

Given an instance $R = \langle S, M \rangle$ of SubsetSum we will construct a binary VCSP instance I_R such that there exists a set of SAC operations on two values a and b for variable X eliminating all SBT occurrences on a and b at X if and only if R is a yes instance.

Let $S = \{a_1, \ldots, a_n\}$. Since we are showing hardness we need only reduce instances for which $n > 1$. The VCSP I_R has $2n + 1$ variables. The domain $D_{2n+1} = \{a, b\}$. All other domains are $\{0, \ldots, 4\}$. The only non-trivial unary constraint is ϕ_{2n+1} where $\phi_{2n+1}(a) = M$ and $\phi_{2n+1}(b) = 0$.

The binary cost function between X_{2n+1} and any other variable depends on whether the index of that variable is even or odd. In all cases there is zero cost if X_{2n+1} is assigned value a. The full table of costs follows. For $i = 1, \ldots, n$, $u \in \{0, \ldots, 4\}$:

$$\begin{aligned}
&\phi_{2i-1,2n+1}(u, a) = 0 && \phi_{2i,2n+1}(u, a) = 0 \\
&\phi_{2i-1,2n+1}(0, b) = 0 && \phi_{2i,2n+1}(0, b) = 0 \\
&\phi_{2i-1,2n+1}(1, b) = 0 && \phi_{2i,2n+1}(1, b) = a_i/2 \\
&\phi_{2i-1,2n+1}(2, b) = a_i/2 && \phi_{2i,2n+1}(2, b) = 0 \\
&\phi_{2i-1,2n+1}(3, b) = a_i/2 && \phi_{2i,2n+1}(3, b) = M + 1 \\
&\phi_{2i-1,2n+1}(4, b) = M + 1 && \phi_{2i,2n+1}(4, b) = a_i/2
\end{aligned}$$

For $i = 1, \ldots, n$ the cost function $\phi_{2i-1,2i}$ is (crisp) equality. That is:

$$\forall u, v \in \{0, \ldots, 4\}, \phi_{2i-1,2i}(u, v) = \begin{cases} 0 & \text{if } u = v \\ \infty & \text{otherwise} \end{cases}$$

All other (binary) cost functions only allow both variables to be zero. That is:

$$\forall u, v \in \{0, \ldots, 4\}, \phi(u, v) = \begin{cases} 0 & \text{if } u = v = 0 \\ \infty & \text{otherwise} \end{cases}$$

Having defined the instance I_R we need to determine each cost q_i to move from the unary cost $\phi_{2n+1}(a)$ to the binary cost function between X_{2n+1} and X_i. After these SAC operations we require that there be no SBT on a and b at X_{2n+1}. We also require that the resultant unary costs of a and b at X_{2n+1} are equal. Hence $\sum_{i=1}^{2n} q_i = M$.

This latter condition induces a constraint on the allowed values of q_i and q_j for every non-infinite allowed pair of values on variables X_i and X_j.

Consider first the crisp constraints which force both X_j and X_k to have value 0. With this X_j and X_k a soft broken triangle can only occur for values $X_j = X_k = 0$. In this case we consider the four costs between values 0 at X_j and X_k and values a and b for X_{2n+1}. A broken triangle does not appear involving X_j and X_k precisely when:

$$(q_j \geq 0 \ \wedge \ q_k \geq 0) \ \vee \ (q_j \leq 0 \ \wedge \ q_k \leq 0) \tag{1}$$

Since $n > 2$ these constraints connect every variable except X_{2n+1}. It follows from Eq. (1) and the fact that $\sum_{i=1}^{2n} q_i = M > 0$, that each $q_i, i = 1 \ldots, q_{2n}$ is non-negative and strictly less than $M + 1$.

Every other constraint that might be involved in a soft broken triangle is a strict equality between a pair of variables X_{2i-1} and X_{2i}. The five non-zero allowed pairs of values in each such constraint give the five following disjunctions.

$$(q_{2i-1} \geq 0 \ \wedge \ q_{2i} \geq 0) \vee (q_{2i-1} \leq 0 \ \wedge \ q_{2i} \leq 0) \tag{2}$$

$$(q_{2i-1} \geq 0 \ \wedge \ q_{2i} \geq a_i/2) \vee (q_{2i-1} \leq 0 \ \wedge \ q_{2i} \leq a_i/2) \tag{3}$$

$$(q_{2i-1} \geq a_i/2 \ \wedge \ q_{2i} \geq 0) \vee (q_{2i-1} \leq a_i/2 \ \wedge \ q_{2i} \leq 0) \tag{4}$$

$$(q_{2i-1} \geq a_i/2 \ \wedge \ q_{2i} \geq M + 1) \vee (q_{2i-1} \leq a_i/2 \ \wedge \ q_{2i} \leq M + 1) \tag{5}$$

$$(q_{2i-1} \geq M + 1 \ \wedge \ q_{2i} \geq a_i/2) \vee (q_{2i-1} \leq M + 1 \ \wedge \ q_{2i} \leq a_i/2) \tag{6}$$

Equation 2 is redundant. Equations 3–6 are equivalent to

$$\text{For } i = 1, \ldots, n, \quad (q_{2i-1} = q_{2i} = 0) \ \vee \ (q_{2i-1} = q_{2i} = a_i/2) \tag{7}$$

Setting $a_i = q_{2i-1} + q_{2i}$, we can see that there exist $q_1, \ldots, q_{2n}$ satisfying the above equations if and only if there is a solution $a_1, \ldots, a_n$ to the SubsetSum instance R.

This reduction is clearly polynomial. Since SubsetSum is NP-complete, we can deduce that testing the existence of a set of soft arc consistency operations on a and b which makes their unary costs equal and eliminates all soft broken triangles, allowing us to apply Proposition 1 and merge a and b, is itself NP-hard.

4 Effect on Search-Tree Size of Merging

It has been shown [7] that in CSPs, BT-merging can increase the number of nodes in the search tree, when arc consistency is maintained during search.

Example 2. Consider an instance I with four boolean variables X_1, X_2, X_3, X_4 and the following constraints $X_1 \Rightarrow X_2$, $\overline{X_1} \Rightarrow X_3$, $X_2 \neq X_3$, $X_2 \neq X_4$, $X_3 \neq X_4$. The two domain values for X_1 can be merged.

A search which assigns $X_1 = 0$ and maintains arc consistency will detect inconsistency:

$$X_1 = 0 \rightarrow X_3 = 1 \rightarrow X_2 = 0 \rightarrow X_4 = 1$$

which wipes out the domain for X_3.

On the other hand, after value merging there are no constraints involving X_1 and inconsistency will only be detected after another variable is instantiated.

However, it has been demonstrated experimentally that BT-merging applied during preprocessing reduces the average number of search-tree nodes by 27% [7].

We can make the following theoretical observation concerning naive (chronological) backtracking.

Proposition 3. *If a search algorithm is used which only prunes nodes based on the cost of the corresponding partial solution and instantiates variables in a fixed order, then merging values due to absence of GASBTs cannot increase the number of nodes visited.*

Proof. Suppose that there is no GASBT on $a, b \in D_i$ and that a, b have been merged to produce a new instance I' in which c is the result of the merge of a and b. Let Y be the variables assigned at some given node of the search tree, where $X_i \in Y$. Consider any assignment s_c to variables $Y \subseteq X$ in I' which assigns c to X_i. Denote by s_a, s_b the assignments which are identical to s_c except that X_i is now assigned a or b (respectively). There were no GASBTs on values a, b in the sub-problem on variables Y. So, from the proof of Proposition 1, we know that the cost of s_c on variables Y is $\min\{cost_Y(s_a), cost_Y(s_b)\}$. Hence, if pruning only depends on this cost, then s_c will only survive (i.e. the corresponding node will not be pruned) in I' if s_a or s_b survives in I. Thus, the total number of nodes cannot increase.

In the special case of binary CSPs, GASBTs are simply broken triangles, which gives us the following corollary.

Corollary 2. *If BT-merging is applied to a CSP then the number of nodes in the search tree cannot increase in a naive backtracking search.*

Of course we can expect that value merging due to the absence of GASBTs will often significantly reduce the number of search nodes visited as it is analogous to BT merging in the CSP, and hence the earlier experiments apply directly.

5 Soft Snakes

The broken triangle is just one example of a forbidden pattern which allows domain reduction. The general notion of forbidden pattern has led to the discovery of several novel value-elimination rules in binary CSPs [2, 6].

In this section we generalise the CSP pattern ∃2snake [2] to VCSPs. This is a further step towards identifying and classifying the generalisation of CSP value-elimination patterns to VCSPS. For simplicity, we concentrate on binary VCSPs, since no general-arity version of ∃2snake has yet been proposed for CSPs.

However, we are very pleased to be able to generalise the snake pattern since it is significantly stronger than GASBT value merging: snake-substitution is a generalisation of a strong form of soft neighbourhood substitution (and hence of weak neighbourhood substitution). Given its relatively low complexity (See Proposition 5) we expect that value merging due the absence of the valued snake pattern will be even more effective than that obtained by the absence of (soft) broken triangles. We first give the definition of soft neighbourhood substitution [5, 13, 17].

Definition 5. *In a binary VCSP instance I, value $a \in D_i$ is* soft neighbourhood substitutable *for $b \in D_i$ if*

$$\phi_i(b) - \phi_i(a) \;+\; \sum_{j \neq i} \min_{c \in D_j} (\phi_{ij}(b,c) - \phi_{ij}(a,c)) \quad \geq \quad 0.$$

Soft neighbourhood substitutability of $a \in D_i$ for $b \in D_i$ implies that b can be replaced in any solution by a. It depends on the costs $\phi_{ij}(a,c)$ and $\phi_{ij}(b,c)$ for all variables X_j. Snake-substitutability extends this by looking at a third variable X_k while allowing a substitution of value $c \in D_j$ by a value $d \in D_k$.

Definition 6. *In a binary VCSP instance I, value $a \in D_i$ is* snake-substitutable *for $b \in D_i$ if*

$$\phi_i(b) - \phi_i(a) \;+\; \sum_{j \neq i} \min_{c \in D_j} (\max_{d \in D_j} f_{ij}(a,b,c,d)) \quad \geq \quad 0$$

where

$$f_{ij}(a,b,c,d) \;=\; \phi_{ij}(b,c) - \phi_{ij}(a,d) + \phi_j(c) - \phi_j(d) + \sum_{k \neq i,j} \min_{e \in D_k} (\phi_{jk}(c,e) - \phi_{jk}(d,e)).$$

The sum in the definition of $f_{ij}(a,b,c,d)$ is the minimum reduction in cost we obtain by replacing c by d (as assignments to variable X_j) in the sub-instance on variables $X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_n$. We can see that snake-substitutability subsumes soft neighbourhood substitutability by setting $c = d$ in Definition 6.

Example 3. Consider a 2-variable instance of MAX-CSP over boolean domains $\{0,1\}$ with a single constraint $X_1 \neq X_2$. In this case, the sum in the definition of $f_{12}(a,b,c,d)$ is zero since there is no third variable X_k such that $k \neq 1,2$. No values are soft neighbourhood substitutable in this instance, but 1 is snake-substitutable for 0 in the domain of X_1 since for any assignment $(0,c)$ to (X_1,X_2) there is an assignment $(1,d)$ of no greater cost.

Proposition 4. *Snake-substitutability is a valid value-elimination rule in binary VCSPs.*

Proof. Consider a binary VCSP instance I in which $a \neq b \in D_n$ and a is snake-substitutable for b. We can assume that I has a solution s_b with $s_b[X_n] = b$ and $cost(s_b) \neq \infty$ otherwise there is nothing to prove.

For each $j \neq n$ let

$$d_j = \arg\max_d \; (f_{nj}(a, b, s_b[X_j], d)) \,.$$

Thus, d_j is, in some sense, the best replacement for $s_b[X_j]$ when we replace b with a at X_n.

Now substitute b for a at X_n in the solution s_b, while also changing assignments to each variable X_j to d_j ($j \neq n$), to obtain the assignment s_a. We only need to show that $cost(s_a) \leq cost(s_b)$, since then s_a is a solution.

Since a is snake substitutable for b at X_n we can use the positivity and finiteness of the expression in Definition 6 to establish that some terms are finite. Finiteness will allow us to use simple subtraction later in the proof.

In Definition 6 we could choose c and e to be the values assigned by s_b to variables X_j and X_k. Finiteness of $cost(s_b)$ then implies that all terms occurring positively in Definition 6 are finite. This in turn implies that all terms occurring negatively in Definition 6 are also finite. So we can see that, for each $j \neq n$ and for each $k \neq n, j$, the following are all finite:

$$\phi_n(a), \phi_{nj}(a, d_j), \phi_j(d_j), \phi_{jk}(d_j, s_b[X_k])$$

We now define n intermediate solutions s_r between s_b and s_a. For $r = 0, \ldots, n-1$,

$$\begin{aligned} s_r[X_j] &= d_j \quad (j = 1, \ldots, r) \\ s_r[X_j] &= s_b[X_j] \quad (j = r+1, \ldots, n-1) \\ s_r[X_n] &= a \end{aligned}$$

Thus, in particular, $s_{n-1} = s_a$ and s_0 is identical to s_b except that variable X_n is assigned the value a.

Let $cost_{n-1}(s)$ denote the cost of assignment s on variables $X_1, \ldots, X_{n-1}$:

$$cost_{n-1}(s) \;=\; \sum_{j=1}^{n-1} \phi_j(s[X_j]) \;+\; \sum_{j=1}^{n-1} \sum_{k=j+1}^{n-1} \phi_{jk}(s[X_j], s[X_k]).$$

From the definition of $cost_{n-1}$, we have:

$$cost(s_b) \;=\; cost_{n-1}(s_0) + \phi_n(b) \;+\; \sum_{r=1}^{n-1} \phi_{rn}(s_b[X_r], b) \tag{8}$$

Then, from the definition of $cost_{n-1}$ and s_r ($r = 0, \ldots, n-1$), and using the finiteness of $\phi_{rk}(d_r, s_r[X_k])$ for $k \neq r, n$ and $\phi_r(d_r)$, we have: for $r = 1, \ldots, n-1$:

$$\begin{aligned} cost_{n-1}(s_{r-1}) \;=\; cost_{n-1}(s_r) \;+\; \sum_{k \neq n,r} &(\phi_{rk}(s_b[X_r], s_r[X_k]) - \phi_{rk}(d_r, s_r[X_k])) \\ &+ \phi_r(s_b[X_r]) - \phi_r(d_r) \end{aligned} \tag{9}$$

We also have

$$cost(s_a) \;=\; cost(s_{n-1}) \;=\; cost_{n-1}(s_{n-1}) \;+\; \phi_n(a) \;+\; \sum_{r=1}^{n-1} \phi_{rn}(d_r, a). \tag{10}$$

We also know that $\phi_n(a) + \sum_{r=1}^{n-1} \phi_{rn}(d_r, a)$ is finite. This allows us to rewrite Eq. (10) as

$$cost_{n-1}(s_{n-1}) = cost(s_a) - \phi_n(a) - \sum_{r=1}^{n-1} \phi_{rn}(d_r, a) \tag{11}$$

By successive substitutions of Eq. (9) ($r = 1, \ldots, n-1$) and then Eq. (11) in Eq. (8), we obtain

$$\begin{aligned}
cost(s_b) = cost(s_a) &+ \phi_n(b) - \phi_n(a) + \sum_{r=1}^{n-1} \{\phi_{rn}(s_b[X_r], b) - \phi_{rn}(d_r, a) \\
&+ \phi_r(s_b[X_r]) - \phi_r(d_r) + \sum_{k \neq n,r} (\phi_{rk}(s_b[X_r], s_r[X_k]) - \phi_{rk}(d_r, s_r[X_k]))\} \\
\geq cost(s_a) &+ \phi_n(b) - \phi_n(a) + \sum_{r=1}^{n-1} \{\phi_{rn}(s_b[X_r], b) - \phi_{rn}(d_r, a) \\
&+ \phi_r(s_b[X_r]) - \phi_r(d_r) + \sum_{k \neq n,r} \min_{e \in D_k} (\phi_{rk}(s_b[X_r], e) - \phi_{rk}(d_r, e))\} \\
= cost(s_a) &+ \phi_n(b) - \phi_n(a) + \sum_{r=1}^{n-1} f_{nr}(a, b, s_b[X_r], d_r) \\
= cost(s_a) &+ \phi_n(b) - \phi_n(a) + \sum_{r=1}^{n-1} \max_{d \in D_r} f_{nr}(a, b, s_b[X_r], d) \\
\geq cost(s_a) &+ \phi_n(b) - \phi_n(a) + \sum_{r=1}^{n-1} \min_{c \in D_r} \max_{d \in D_r} f_{nr}(a, b, c, d) \\
\geq cost(s_a)
\end{aligned}$$

by definition of snake substitutability. Hence, given any solution which assigns b to X_n, we can find a solution which assigns a to X_n.

Example 4. Consider a binary VCSP instance I in which there is a crisp equality constraint $X_h = X_i$ between variables X_h and X_i which have identical domains and $|D_i| > 1$. Clearly, we could merge these two variables to form a single variable Y. Suppose that a is (weak) neighbourhood substitutable for b at Y in this new variable-merged instance $\overline{I}$. However, because of the crisp equality constraint, a is not (weak) neighbourhood substitutable for b in I. Nonetheless, a is snake-substitutable for b in I. To see this, for all $j \neq i, h$, set $d = c$ and for $j = h$, set $d = a$ in Definition 6. Then the snake-substitutability of a for b in I follows from the (weak) neighbourhood substitutability of a for b in $\overline{I}$.

We now analyse the computational complexity of checking that no values are snake substitutable (according to Definition 6). Direct application of the definition leads to a time complexity of $O(n^3d^3 + n^2d^4)$ and a space complexity of $O(n^2d^2)$. However, we can improve this complexity in the case of finite costs.

In this case, we can rewrite the definition of $f_{ij}(a,b,c,d)$ as follows:

$$\begin{aligned} f_{ij}(a,b,c,d) &= \phi_{ij}(b,c) - \phi_{ij}(a,d) + \phi_r(c) - \phi_r(d) \\ &\quad + \sum_{k\neq i,j} \min_{e\in D_k} (\phi_{jk}(c,e) - \phi_{jk}(d,e)) \\ &= \phi_{ij}(b,c) - \phi_{ij}(a,d) + \phi_r(c) - \phi_r(d) \\ &\quad - \min_{e\in D_i} (\phi_{ji}(c,e) - \phi_{ji}(d,e)) \;+\; \sum_{k\neq j} \min_{e\in D_k} (\phi_{jk}(c,e) - \phi_{jk}(d,e)) \end{aligned}$$

which allows us to check that no values are snake substitutable in time complexity $O(n^2d^4)$.

Proposition 5. *In finite-valued VCSPs, snake substitutable values can be found in time complexity $O(n^2d^4)$.*

It is worth pointing out that this is only a factor of d greater than the complexity of checking that no soft neighbourhood substitutions are possible [5].

6 Conclusion

We have extended the notion of broken triangle from CSPs to VCSPs. This has allowed us to define a valid domain-reduction operation based on value-merging. We have extended the usefulness of this pattern by considering SAC operations that might enable us to perform extra reductions. In each case we have briefly considered the complexity of the reduction. We have shown that it is NP-hard to determine whether there exists some set of SAC operations that can allow us to perform extra GASBT reductions. However this is a reduction from SubsetSum and it is well known that there are pseudo-polynomial algorithms that solve this problem. It is an open question to determine whether there is a pseudo-polynomial algorithm for finding a set of SAC operations that allow us to merge values. It seems unlikely as this problem is in fact a quadratic optimisation problem.

In the final section we considered another domain-reduction operation called snake-substitutability which is a strong generalisation of neighbourhood substitutability in the case of binary VCSPs. From a theoretical point of view, SBT-merging and snake-substitutability are incomparable, since there are instances where one can be applied but not the other, even if on binary finite-valued instances, snake-substitutability subsumes SBT-merging. It is an ongoing research program to discover a general rule or classification of all domain reduction patterns for the VCSP.

Acknowledgements. We would like to thank Wady Naanaa for useful discussion concerning the generalisation to the general-arity case.

References

1. Alfonsín, J.L.R.: On variations of the subset sum problem. Discrete Appl. Math. **81**(1–3), 1–7 (1998)
2. Cohen, D.A., Cooper, M.C., Escamocher, G., Zivny, S.: Variable and value elimination in binary constraint satisfaction via forbidden patterns. J. Comput. Syst. Sci. **81**(7), 1127–1143 (2015)
3. Cohen, D.A., Jeavons, P., Jefferson, C., Petrie, K.E., Smith, B.M.: Symmetry definitions for constraint satisfaction problems. Constraints **11**(2–3), 115–137 (2006)
4. Cooper, M.C.: An optimal k-consistency algorithm. Artif. Intell. **41**(1), 89–95 (1989)
5. Cooper, M.C.: Reduction operations in fuzzy or valued constraint satisfaction. Fuzzy Sets Syst. **134**(3), 311–342 (2003)
6. Cooper, M.C.: Beyond consistency and substitutability. In: O'Sullivan, B. (ed.) CP 2014. LNCS, vol. 8656, pp. 256–271. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10428-7_20
7. Cooper, M.C., Duchein, A., Mouelhi, A.E., Escamocher, G., Terrioux, C., Zanuttini, B.: Broken triangles: from value merging to a tractable class of general-arity constraint satisfaction problems. Artif. Intell. **234**, 196–218 (2016)
8. Cooper, M., de Givry, S., Sanchez, M., Schiex, T., Zytnicki, M., Werner, T.: Soft arc consistency revisited. Artif. Intell. **174**(7), 449–478 (2010)
9. Freuder, E.C.: Eliminating interchangeable values in constraint satisfaction problems. In: Dean, T.L., McKeown, K. (eds.) Proceedings of the 9th National Conference on Artificial Intelligence, Anaheim, CA, USA, 14–19 July 1991, vol. 1, pp. 227–233. AAAI Press/The MIT Press (1991)
10. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York (1990)
11. Gent, I.P., Petrie, K.E., Puget, J.F.: Symmetry in constraint programming. In: Rossi, F., van Beek, P., Walsh, T. (eds.) Handbook of Constraint Programming, Foundations of Artificial Intelligence, vol. 2, pp. 329–376. Elsevier (2006)
12. de Givry, S., Prestwich, S.D., O'Sullivan, B.: Dead-end elimination for weighted CSP. In: Schulte, C. (ed.) CP 2013. LNCS, vol. 8124, pp. 263–272. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40627-0_22
13. Goldstein, R.: Efficient rotamer elimination applied to protein side-chains and related spin glasses. Biophys. J. **66**(5), 1335–1340 (1994)
14. Guerinik, N., Van Caneghem, M.: Solving crew scheduling problems by constraint programming. In: Montanari, U., Rossi, F. (eds.) CP 1995. LNCS, vol. 976, pp. 481–498. Springer, Heidelberg (1995). https://doi.org/10.1007/3-540-60299-2_29
15. van Hoeve, W.J., Katriel, I.: Global constraints. In: Rossi, F., van Beek, P., Walsh, T. (eds.) Handbook of Constraint Programming, Foundations of Artificial Intelligence, vol. 2, pp. 169–208. Elsevier (2006)
16. Jeavons, P., Cohen, D., Cooper, M.: A substitution operation for constraints. In: Borning, A. (ed.) PPCP 1994. LNCS, vol. 874, pp. 1–9. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-58601-6_85
17. Lecoutre, C., Roussel, O., Dehani, D.E.: WCSP integration of soft neighborhood substitutability. In: Milano, M. (ed.) CP 2012. LNCS, pp. 406–421. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33558-7_31
18. Likitvivatanavong, C., Yap, R.H.C.: Eliminating redundancy in CSPs through merging and subsumption of domain values. SIGAPP Appl. Comput. Rev. **13**(2), 20–29 (2013)

19. Roney-Dougal, C.M., Gent, I.P., Kelsey, T., Linton, S.: Tractable symmetry breaking using restricted search trees. In: Proceedings of the 16th European Conference on Artificial Intelligence, ECAI 2004, pp. 211–215 (2004)
20. Senkul, P., Toroslu, I.H.: An architecture for workflow scheduling under resource allocation constraints. Inf. Syst. **30**(5), 399–422 (2005)
21. Smith, B.M., Bistarelli, S., O'Sullivan, B.: Constraint symmetry for the soft CSP. In: Bessière, C. (ed.) CP 2007. LNCS, vol. 4741, pp. 872–879. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74970-7_66