Technical Communications of the International Conference on Logic Programming, 2010 (Edinburgh), pp. 114–123
http://www.floc-conference.org/ICLP-home.html

CONTRACTIBILITY AND CONTRACTIBLE APPROXIMATIONS OF SOFT GLOBAL CONSTRAINTS

MICHAEL J. MAHER [1,2]

[1] NICTA, Locked Bag 6016, The University of New South Wales, Sydney, NSW 1466, Australia

[2] School of Computer Science and Engineering, The University of New South Wales, Sydney, NSW 2052, Australia
E-mail address: michael.maher@nicta.com.au

Abstract. We study contractibility and its approximation for two very general classes of soft global constraints. We introduce a general formulation of decomposition-based soft constraints and provide a sufficient condition for contractibility and an approach to approximation. For edit-based soft constraints, we establish that the tightest contractible approximation cannot be expressed in edit-based terms, in general.

1. Introduction

Soft constraints are useful for addressing problems that might be overconstrained. Open global constraints [1, 11, 14] allow variables to be added to the global constraint during execution, which is vital when we want to interleave problem construction and problem solving. Contractible approximations of global constraints are a necessary part of implementing open versions of the constraint. In this paper we investigate contractibility [13] and contractible approximations [14] of soft constraints in the sense of Petit *et al* [18] for the purpose of implementing versions that are dynamic, or open, in the sense of Barták [1]. Contractibility of soft constraints was studied in [15] but approximations have not been investigated.

We investigate two general classes of soft constraints based, respectively, on decompositions and edit-distance. We improve on several results of [15] and establish new results for these classes. While hard constraints seem to be amenable to tight contractible approximation, at least in the cases studied so far [14], we show that soft constraints are much less so.

Section 2 provides some preliminaries on open global constraints and contractibility. In section 3 we introduce a very general class of decomposition-based soft constraints, repeat the definition of edit-based soft constraints from [15] and explore some consequences of

1998 ACM Subject Classification: D.1.6 Logic Programming; D.3.2 Language Classifications, Constraint and logic languages; D.3.3 Language Constructs and Features, Constraints .

Key words and phrases: constraint logic programming, global constraints, open constraints, soft constraints.

NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

Technical Communications of the 26th International Conference on Logic Programming, Edinburgh, July, 2010
Editors: Manuel Hermenegildo, Torsten Schaub
LIPIcs - Leibniz International Proceedings in Informatics. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Germany
Digital Object Identifier: 10.4230/LIPIcs.ICLP.2010.114

these definitions. Section 4 investigates contractible approximations for soft constraints in these two classes.

2. Background

The reader is assumed to have a basic knowledge of constraint logic programming, CSPs, global constraints, and filtering, as might be found in [9, 19, 2].

For the purposes of this paper, a global constraint is a relation over a single sequence of variables. Other arguments of a constraint are considered parameters and are assumed to be fixed before execution. A sequence of variables will be denoted by $\vec{X}$ or $[X_1, \ldots, X_n]$. We make no *a priori* restriction on the variables that may participate in the sequence except that, in common with most work on global constraints, we assume that no variable appears more than once in a single constraint.

We assume that each use of a global constraint has a static type T that assigns, for every position i of its argument, a set of values. Thus every variable X_i in $\vec{X}$ has a static type $T(X_i)$ of values that it may take. This is distinct from the *domain* $D(X)$ of a variable X, which changes during execution. We always have $D(X) \subseteq T(X)$.

We formalize the semantics of a global constraint C as a formal language L_C. A word $d_1 d_2 \ldots d_n$ appears in L_C iff the constraint $C([X_1, X_2, \ldots, X_n])$ has a solution $X_1 = d_1, \ldots X_n = d_n$. Thus, for example, the semantics of AllDifferent is $\{a_1 \ldots a_n \mid \forall i, j\ i \neq j \rightarrow a_i \neq a_j, n \in \mathbb{N}\}$ and the semantics of Regular$(\mathcal{A}, \vec{X})$ is $L(\mathcal{A})$, the language accepted by $\mathcal{A}$. When it is convenient, we will describe languages with Kleene regular expressions.

We will need the following definitions later. Let $P(L) = \{w \mid \exists u\ wu \in L\}$ denote the set of prefixes of a language L, called the prefix-closure of L. We say L is *prefix-closed* if $P(L) = L$.

Constraint logic programming supports the generation of new variables and new constraints during execution. Use of open constraints pre-supposes additional capabilities to add variables to an open constraint and to close an open constraint. In this paper we will abstract away the details of the language mechanisms that provide these capabilities so that we can focus on the filtering/propagation for open constraints.

There are three models of open constraint that have been proposed [1, 11, 14]. In this paper we follow the model of Barták [1] as refined in [13]: the collection of variables forms a sequence, to which variables may be added at the right-hand end only.

We take *filtering* to refer to any function f that reduces domains, that is, $\forall X\ f(D)(X) \subseteq D(X)$. A filtering algorithm f for a constraint C is *sound* if every solution of C in D also appears in $f(D)$. We say a domain D *defines an assignment* if $\forall X\ |D(X)| = 1$; in that case the assignment maps each X to the element of $D(X)$. We say filtering performs *complete checking* if, whenever D defines an assignment, the result of filtering with a constraint C is D iff the assignment satisfies C. Soundness and complete checking can be considered minimal requirements for filtering methods [20].

Consistency conditions like domain consistency are inappropriate for open constraints because some of the variables in an open constraint will be unspecified during part of the execution. The counterpart of domain consistency is open D-consistency, defined in [14].

During execution, we may extend an open constraint $C(\vec{X})$ with an extra variable Y to $C(\vec{X}Y)$. We would like to do filtering on the smaller constraint without knowing whether it will be extended by Y, or further, and without creating a choicepoint. When we can do

this, we have a kind of monotonicity property of C, called contractibility [13], which can be characterized as follows.

Definition 2.1. We say a constraint $C(\vec{X})$ is *contractible* iff L_C is prefix-closed.

Any filtering method satisfying the minimal requirements discussed above requires contractibility to guarantee that closed filtering is sound for an open constraint. This refines a result of [13].

Theorem 2.2. *Let C be a constraint, and consider a sound filtering method that performs complete checking. It is always sound to interleave filtering and the addition of new variables iff C is contractible.*

Consequently, for contractible constraints, filtering does not need to be undone if the list is lengthened. That is, algorithms for filtering a closed contractible constraint are valid also for the corresponding open constraint.

Conversely, any constraint C that is not contractible might need to undo the effects of filtering if the list is lengthened. Barták [1] proposed an implementation approach to avoid this effect. Essentially, a propagator for a contractible approximation of C is used until C is closed, when a propagator for C is used. The importance of contractible approximations lies in this ability to provide filtering for open uncontractible constraints. Tight approximations can yield best-possible filtering [14].

Theorem 2.3. *Let C_{app} be the tightest contractible approximation to C, and suppose we have closed propagators for C_{app} and C that maintain domain consistency wrt $\vec{X}$. Then Barták's proposal maintains open D-consistency for C.*

3. Contractibility of Soft Constraints

We consider "soft" global constraints in the style of [18]. In such constraints there is a violation measure, which measures the degree to which an assignment to the variables violates the associated "hard" constraint, and solutions are assignments that satisfy an upper bound on the violation measure. Thus such soft constraints have the form $m(\vec{X}) \leq Z$, where m is the violation measure[1]. We refer to the hard constraint as $C(\vec{X})$ and the corresponding soft constraint as $C_s(\vec{X}, Z)$.

We say that a function f is an *accumulation function* if it maps sequences of values to a single value. We say f is *non-decreasing* if for every sequence of values $\vec{X}$ and value Y, $f(\vec{X}) \leq f(\vec{X}Y)$. Addressing the contractability of such constraints is made easier by the following characterization [13].

Proposition 3.1. *A soft constraint $m(\vec{X}) \leq Z$ is contractible iff m is a non-decreasing accumulation function.*

Consequently, we will also call such functions m contractible. Thus, to evaluate whether or not soft constraints are contractible we must consider the form of the violation measure, and whether it forms a contractible function.

[1] Also called violation cost [18].

Definition 3.2. A *violation measure* for a sublanguage L of a language L' is a function m which maps L' to the non-negative real numbers, such that if $w \in L$ then $m(w) = 0$. m is *proper* for L if for all words $w \in L'$, $m(w) = 0$ iff $w \in L$. A violation measure for a constraint $C(\vec{X})$ is a violation measure for L_C as a sublanguage of the static type $T(\vec{X})$.

For example, a use of AllDifferent might give the set $\mathbb{Z}$ of integers as the static type of each variable. A violation measure might then be the number of disequalities $X_i \neq X_j, i \neq j$ violated by a valuation for $\vec{X}$, or the number of variables equal to another variable, or the minimum absolute value of the sum over i of values c_i such that $\forall j\ j \neq i \rightarrow X_i + c_i \neq X_j$. It is easy to see that each of these defines a violation measure. The third is not a proper violation measure because, for example, the word 11233 gives rise to values of c_i of $-1, -1, 0, 1, 1$. Thus $m(11233) = 0$ but $11233 \notin L_C$.

Proper violation measures for a language L are a refinement of the characteristic function of L, and can be considered more intuitive than non-proper measures. Most violation measures in the literature are proper for their intended language. Although any function from words to non-negative reals can be considered a proper violation measure by appropriate choice of language L, in practice the hard constraint determines L and the violation measure is then designed to be proper. A non-proper measure can be considered misleading because a word w that violates the language L can have a violation measure of 0. We admit non-proper violation measures mainly because contractible approximations considered in Section 4 can be non-proper.

There are three broad classes of violation measures [15]: those based on constraint decomposition, edit distance, and graph properties. We address the first two classes in the following subsections. The richness of the graph property framework [3] makes it difficult to obtain general results on contractibility.

3.1. Decomposition-based Violation Measures

Many hard constraints can be decomposed into elementary constraints, whether naturally (such as the decomposition of AllDifferent into disequalities) or by design, as in [5]. Violation measures can be constructed by combining the violations of each elementary constraint. We define a general class of decomposition-based violation measures that includes as special cases: primal graph based violation costs [18], decomposition-based violation measures of [10], the value-based violation measure for GCC [18, 10], the measures used for the soft Sequence constraint [16] and the soft Cumulative constraint [17], and the class of decomposition-based measures discussed in [15]. We begin with several definitions.

A *weighted set* is a pair (S, w) where S is a set and w is a function mapping each element of S to a non-negative real number or ∞. Values not in S have weight 0. If these are the only values of weight 0 we say (S, w) is *proper*. A weighted set is a minor generalization of a multiset. A weighted set (S_1, w_1) is a *sub-weighted set* of weighted set (S_2, w_2) if, for every element $s \in S_1$, $w_1(s) \leq w_2(s)$. Union of weighted sets is defined by $(S_1, w_1) \cup (S_2, w_2) = (S_1 \cup S_2, w_1 + w_2)$. When a weighted set contains expressions with variables that are subject to substitution, the application of a substitution might unify elements of the set. Hence, $(S, w)\theta$ denotes $(S\theta, w')$ where $w'(s)$ is the sum of $w_1(s')$ over all $s' \in S$ such that $s'\theta \equiv s$.

We need to carefully formalize the notion of decomposition. Decomposition is a function that maps a constraint C with a given type T and a sequence of variables $\vec{X}$ to a tuple

$(\vec{X}, \vec{U}, T', S, w)$ where T' is an extension of T, $\vec{U}$ is a collection of new variables, $T(\vec{U})$ is their corresponding types, and (S, w) is a proper weighted set of elementary constraints over $\vec{X}\vec{U}$ such that $C(\vec{X}) \leftrightarrow \exists \vec{U}\ T'(\vec{U}) \wedge \bigwedge_{s \in S} s$. The weights are used only to emphasize some constraints in a decomposition over others; in particular, the infinite weight allows us to specify elementary constraints that must not be violated. An *unweighted decomposition* is one where all constraints in S have the same, non-zero weight. In that case, we may omit w. We write DECOMP$(C(\vec{X}))$ to express the weighted set (S, w), or simply S when the decomposition is unweighted. This definition of decomposition is very broad, perhaps too broad, since it allows the set of elementary constraints to vary radically as the length of $\vec{X}$ changes.

An *error function* e maps an elementary constraint and a valuation to a non-negative real number, representing the amount of error (or violation) of the constraint by the valuation. We require that $e(v, c) = 0$ iff c is satisfied by v. We extend e to weighted sets of constraints by defining $e(v, (S, w)) = (S', w')$ where $S' = \{e(v, s) \mid s \in S\}$ and $w'(x) = \sum_{v(s)=x} w(s)$.

A *combining function* maps a set of numbers and a weighting function to a single number. A combining function *comb* is *monotonic* if, whenever (S_1, w_1) is a sub-weighted set of (S_2, w_2), $comb(S_1, w_1) \leq comb(S_2, w_2)$. The function *comb* is *disjunctive* if for all weighted sets of reals (S, w), $comb(S, w) = 0$ iff $S = \{0\}$. Counting non-zero values, summation, sum of squares, and maximization are examples of monotonic, disjunctive combining functions; product and minimization are neither monotonic nor disjunctive.

Definition 3.3. A *decomposition-based violation measure* m for a constraint $C(\vec{X})$ with type T is based on a decomposition $(\vec{X}, \vec{U}, T', S, w)$ of $C(\vec{X})$, an error function e, and a combining function *comb* and is defined by, for each valuation v of $\vec{X}$,

$$m(C(v(\vec{X}))) = \min_{v'} comb(e(v', \text{DECOMP}(C(\vec{X}))))$$

where we minimize over all extensions v' of v to $\vec{U}$ that satisfy T'.

This definition was inspired by the formulation of hierarchical constraints in [6, 7]. The decomposition measures of [18, 10] can be obtained when the error function $e(v, c)$ returns 0 if v satisfies c and 1 otherwise, and the combining function is summation. The value-based measures of [18, 10, 16, 17] also use summation as the combining function, but use an error function that returns the amount by which the constraint c is violated by the valuation v. If we use maximization or the sum of squares in place of summation we have new violation measures similar to the *worst-case-better* and *least-squares-better* comparators of [6, 7]. Clearly many violation measures are available for a constraint by making different choices for the decomposition and the error and combining functions.

There is a powerful sufficient condition for a decomposition-based violation measure to be proper.

Proposition 3.4. *Let m be a decomposition-based violation measure for a constraint C. If comb is disjunctive then m is proper for L_C.*

We say that one formula $(\vec{X}, \vec{U}, T_1, S_1, w_1)$ is *covered* by another formula $(\vec{W}, \vec{V}, T_2, S_2, w_2)$ if there is a substitution θ that maps $\vec{X}$ into $\vec{W}$ and $\vec{U}$ into $\vec{V} \cup \vec{W} \cup \Sigma$, where Σ is a set of constants, such that $T_1(\vec{X}) = T_2(\vec{X}\theta)$, $(S_1, w_1)\theta$ is a sub-weighted set of (S_2, w_2) and $T_2(\vec{U}\theta) \subseteq T_1(\vec{U})$.

Example 3.5. The decomposition of $\textsc{AllDifferent}(\vec{X})$ into a set of disequalities is formalized as $(\vec{X}, \emptyset, T, S, w)$ where S is the set of disequalities and w gives every disequality a weight of 1. It is clear that the decomposition of $\textsc{AllDifferent}(\vec{X})$ is covered by that of $\textsc{AllDifferent}(\vec{X}Y)$ where the substitution is the identity.

$\textsc{Contiguity}$ is implemented in [12] essentially by the decomposition

$$\textsc{Contiguity}(\vec{X}) \leftrightarrow \exists \vec{L}, \vec{R}, X_0, X_{n+1} \bigwedge_{i=1}^{n} C'(X_{i-1}, R_{i-1}, L_i, X_i, R_i, L_{i+1}, X_{i+1})$$

for a constraint C'. This decomposition is formalized as $(\vec{X}, \vec{L}\vec{R}X_0X_{n+1}, T, S, w)$ where T gives all variables a type of $\{0, 1\}$, S is the set of C' constraints, and w gives every constraint a weight of 1. Alternatively, if contiguity is more important for variables nearer the right end of the sequence $\vec{X}$, we might weight each C' constraint by the largest index of a variable appearing in it. The decomposition of $\textsc{Contiguity}(\vec{X}Y)$ covers that of $\textsc{Contiguity}(\vec{X})$ where the substitution is the identity on $\vec{X}$, $\vec{L}$, and $\vec{R}$.

We can now provide a sufficient condition for a soft constraint with a decomposition-based violation measure to be contractible.

Proposition 3.6. *Let C_s be a soft constraint with a decomposition-based violation measure defined using a monotonic combining function. Let $(\vec{X}, \vec{U}, T_1, S_1, w_1)$ be the decomposition of $C(\vec{X})$ and $(\vec{X}Y, \vec{V}, T_2, S_2, w_2)$ be the decomposition of $C(\vec{X}Y)$. If $(\vec{X}, \vec{U}, T_1, S_1, w_1)$ is covered by $(\vec{X}Y, \vec{V}, T_2, S_2, w_2)$ via a substitution that is the identity on $\vec{X}$ then C_s is contractible.*

It follows that the constraints in Example 3.5 are contractible. Covering is only a sufficient condition for contractibility, as the following example demonstrates.

Example 3.7. Consider the definition of a rising sawtooth relation rs on variables $\vec{X}$. In such a relation, the subsequence of values in even numbered positions forms a non-decreasing sequence, and every value in odd numbered positions is greater than its immediately adjacent neighbours. This relation can be decomposed into elementary constraints as follows. The decomposition is defined recursively, but notably requires two recursive cases, corresponding to the distinction between odd and even length sequences.

$$\begin{aligned}
&\textsc{decomp}(rs([])) &&= \mathit{true}\\
&\textsc{decomp}(rs([X_1])) &&= \mathit{true}\\
&\textsc{decomp}(rs([X_1, X_2])) &&= X_1 \geq X_2\\
&\textsc{decomp}(rs([X_1, \ldots, X_{2n}, X_{2n+1}])) &&=\\
&\qquad \textsc{decomp}(rs([X_1, \ldots, X_{2n}])) \wedge X_{2n+1} \geq X_{2n}\\
&\textsc{decomp}(rs([X_1, \ldots, X_{2n}, X_{2n+1}, X_{2n+2}])) &&=\\
&\qquad \textsc{decomp}(rs([X_1, \ldots, X_{2n}])) \wedge X_{2n+1} \geq X_{2n+2} \wedge X_{2n+2} \geq X_{2n}
\end{aligned}$$

Consider the soft constraint derived from this decomposition by counting the number of violations. It is clear that the sufficient condition of Proposition 3.6 does not apply because there is no covering. Nevertheless, we can verify that a decomposition-based soft rs constraint is contractible. Note first that when $\vec{X}$ has even length $\textsc{decomp}(rs(\vec{X})) \subseteq \textsc{decomp}(rs(\vec{X}Y))$ and consequently the violation measure is non-decreasing in this case. When $\vec{X}$ has odd length the relationship is less obvious. However, we know that

$$\neg(X_{2n+1} \geq X_{2n}) \rightarrow \neg(X_{2n+1} \geq X_{2n+2}) \vee \neg(X_{2n+2} \geq X_{2n})$$

Hence, any valuation for the variables that gives rise to a violation of $X_{2n+1} \geq X_{2n}$ will also give rise to a violation of $X_{2n+1} \geq X_{2n+2}$, or $X_{2n+2} \geq X_{2n}$, or both. Thus the violation measure is non-decreasing in this case also. Since the violation measure is non-decreasing, the decomposition-based soft rs constraint is contractible.

Similarly, the violation measures derived from summing the amount of violation or taking the maximum amount of violation of any elementary constraint lead to contractible soft rs constraints.

This example demonstrates a major limitation of the sufficient condition in Proposition 3.6: it addresses only the syntactic structure of the decomposition. However some constraints, such as rs, require reasoning about the semantics of the elementary constraints in order to recognise that the decomposition-based soft constraint is contractible. (For rs we exploited the knowledge that $\geq$ forms a total order.)

3.2. Edit-based Violation Measures

The *edit-based* violation measures use a notion of edit distance, which is the minimum number of edit operations required to transform a word into a word of L_C. There are many possible edit operations but the common ones are: to substitute one letter for another, to insert a letter, to delete a letter, and to transpose two adjacent letters[2]. This class includes the *variable-based* violation measures [18, 10], the *object-based* measures of [3], and edit-based measures from [10].

Let $\alpha, \beta, \gamma, \delta$ be non-negative weights for the edit operations substitution, insertion, deletion and transposition, respectively, and let n_s, n_i, n_d, n_t be the number of the respective operations used in an edit. Then, for any language L, we define $m_L(w) = \min_{edits} \alpha n_s + \beta n_i + \gamma n_d + \delta n_t$ to be the minimum, over all edits that transform the word w to an element of L, of the weighted sum of the edit operations.

Definition 3.8. An *open edit-based violation measure* for a sublanguage L of L' is a weighted edit distance $m_{P(L)}$ for $P(L)$. An open edit-based violation measure m for L is *proper* if $m(w) = 0$ iff $w \in P(L)$ for every $w \in L'$.

We can characterize when an open edit-based violation measure is proper. Roughly, m is improper iff some edits have zero cost and these are able to edit some $w \in L' \backslash P(L)$ to $w' \in P(L)$.

Proposition 3.9. *Let m be an open edit-based violation measure for L where $P(L)$ is a sublanguage of L', with weights α, β, γ and δ.*

m is proper iff one of the following conditions holds:

- $\min\{\alpha, \beta, \gamma, \delta\} > 0$
- $\alpha = 0$, $\min\{\beta, \gamma\} > 0$ *and* $L' \cap SameLength(P(L)) \subseteq P(L)$
- $\beta = 0$, $\min\{\alpha, \gamma, \delta\} > 0$ *and* $L' \cap SubSeq(P(L)) \subseteq P(L)$
- $\gamma = 0$ *and* $L' \subseteq P(L)$
- $\delta = 0$, $\min\{\alpha, \beta, \gamma\} > 0$ *and* $L' \cap Perm(P(L)) \subseteq P(L)$
- $\alpha = \beta = 0$, $\gamma > 0$ *and* $L' \subseteq Shorter(P(L))$
- $\beta = \delta = 0$, $\min\{\alpha, \gamma\} > 0$ *and* $L' \cap Subset(P(L)) \subseteq P(L)$

[2]Edit distance based on these operations is known as Damerau-Levenshtein distance.

where, for any language L,
$SameLength(L)$ is the set of all words of the same length as a word of L,
$Shorter(L)$ is the set of all words the same length or shorter than a word of L,
$Perm(L)$ is the set of all permutations of words of L,
$SubSeq(L)$ is the set of all subsequences of a word of L, and
$Subset(L)$ is set of all words whose letters form a submultiset of the letters of a word of L.

In many cases, edit-based violation measures are contractible. This is a slight strengthening of a theorem of [15].

Theorem 3.10. *Let C_s be a soft constraint with an open edit-based violation measure. Suppose* $\min\{\alpha, \beta, \gamma\} \leq \delta$.

Then C_s is contractible.

An example from [15] shows that if $\delta < \min\{\alpha, \beta, \gamma\}$ then an edit-based soft constraint might be uncontractible. Thus Theorem 3.10 cannot be strengthened further without imposing extra conditions on C_s.

4. Contractible Approximations of Soft Constraints

As with hard constraints, when a soft constraint is uncontractible we can use a contractible approximation while the constraint is open.

We reformulate the notion of tight approximation for soft constraints of the form $m(\vec{X}) \leq Z$ as follows. A violation measure m_1 is an *approximation* of the violation measure m if, for all words $\vec{a}$, $m_1(\vec{a}) \leq m(\vec{a})$. We order violation measures with the pointwise extension of the ordering on the reals: $m_1 \leq m_2$ iff $\forall \vec{a}\ m_1(\vec{a}) \leq m_2(\vec{a})$. A contractible approximation m_1 to a violation measure m is *tight* if, for all contractible functions m_2, if $m_1 \leq m_2 \leq m$ then $m_2 = m_1$. Given two contractible approximations m_1 and m_2 to a violation measure m, we say m_2 is *tighter* than m_1 if $m_1 \leq m_2$. We write m^* to denote the tightest contractible approximation of m.

We can characterize the tightest contractible approximation of a violation measure, independent of how the violation measure is formulated.

Proposition 4.1. *Let m be a violation measure. The tightest contractible approximation to m is characterized by $m^*(\vec{a}) = \inf_{\vec{b}} m(\vec{a}\vec{b})$, where the infimum is taken over all finite sequences $\vec{b}$.*

4.1. Decomposition-based Violation Measures

One way to obtain a contractible approximation is to ignore parts of a decomposition that cause incontractibility. A *weakening* of a decomposition of a constraint $C(\vec{X})$ is a function that, for every sequence $\vec{X}$, maps the decomposition $(\vec{X}, \vec{U}, T', S, w)$ to $(\vec{X}, \vec{U}, T', S', w')$ where (S', w') is a sub-weighted set of (S, w). For this weakened decomposition we can apply the sufficient condition of Proposition 3.6.

Proposition 4.2. *Consider a decomposition-based violation measure m for a constraint $C(\vec{X})$ and a weakening W of the decomposition. Suppose m is defined via a monotonic combining function. If, for every sequence $\vec{X}$, the weakening of the decomposition of $C(\vec{X})$ is covered by the weakening of the decomposition of $C(\vec{X}Y)$ via a substitution that is the*

identity on $\vec{X}$ then the measure m' defined by using the weakened decompositions is a contractible approximation of m.

This result shows an approach to finding a contractible approximation to $C(\vec{X})$, but it provides no guarantee of finding a useful approximation. Like Proposition 3.6, it follows a syntactic approach and has the same inherent weakness.

4.2. Edit-based Violation Measures

Recall that an edit-based violation measure m is contractible if $\delta \geq \min\{\alpha, \beta, \gamma\}$ (Theorem 3.10). If $\delta < \min\{\alpha, \beta, \gamma\}$ then m might be uncontractible and we must consider contractible approximations. We can provide generic contractible approximations for edit-based soft constraints by modifying the weights to accord with the sufficient condition of Theorem 3.10.

Proposition 4.3. *Let m be an open edit-based violation measure for a constraint C with weights $\alpha, \beta, \gamma, \delta$ where $\delta < \min\{\alpha, \beta, \gamma\}$. Then the following violation measures are contractible approximations of m for C.*

(1) *m_1 based on weights $\delta, \beta, \gamma, \delta$*
(2) *m_2 based on weights $\alpha, \delta, \gamma, \delta$*
(3) *m_3 based on weights $\alpha, \beta, \delta, \delta$*
(4) *m_4 defined by $m_4(w) = \max\{m_1(w), m_2(w), m_3(w)\}$*

Clearly m_4 is the tightest of these approximations, and might be sufficient in practice. However, in general, this approximation is not tight, as the following example shows.

Example 4.4. Let $L = (abc)^*$, so that $P(L) = L \cup La \cup Lab$. Let $\alpha = \beta = \gamma = 4$ and $\delta = 1$. Consider $w = bbb(abc)^3ca$. Two kinds of edits are needed, addressing the initial b's and the trailing ca. Then $m(w) = 12$ from substituting for the first and third b, and deleting the last c. $m(wb) = 10$ using the same substitutions and two transpositions on c. Thus m is not contractible.

The tightest approximation to m has $m^*(w) = 10$. Now consider the approximations in Proposition 4.3. If we reduce α to 1 then $m_1(w) = 4$ by applying four substitutions. If we reduce β to 1 then $m_2(w) = 8$ by inserting a and c around each initial b and inserting ab before the last c. If we reduce γ to 1 then $m_3(w) = 4$ by deleting the three b's and the last c. Thus $m_4(w) = 8$.

This shows that m_4 is not the tightest contractible approximation to m, since $m_4(w) \neq m^*(w)$.

The question now arises: how to express m^* in edit-based terms so that a closed propagator for $m(\vec{X}) \leq Z$ might be adapted to implement $m^*(\vec{X}) \leq Z$. Disappointingly, this turns out to be impossible, in general.

Theorem 4.5. *There is an open edit-based violation measure m for a language L such that its tightest contractible approximation cannot be expressed as a proper (not necessarily open) edit-based violation measure on any language.*

The language and violation measure demonstrating this claim are those from Example 4.4. Given that the language is so simple, we can expect that many uncontractible edit-based violation measures cannot be tightly approximated by a contractible edit-based violation measure. This contrasts markedly with work on hard constraints, where tight contractible

approximations of several uncontractible hard constraints can be formulated in terms of the original hard constraint [14, 15]. It suggests that the edit-based implementation of the closed constraint is not a suitable basis for implementing the tight approximation.

5. Conclusions

We have investigated violation measures for soft constraints based on decomposition and edit-distance. We defined a class of decomposition-based violation measures that generalizes several previous works. For both forms of violation measures we identified sufficient conditions for constraints to be proper and strengthened results of [15] on when they are contractible. Finally, we found that the edit-based framework is not expressive enough to represent tight contractible approximations, in general.

Acknowledgements

Thanks to the referees for their comments, which helped improve this paper.

References

[1] R. Barták, Dynamic Global Constraints in Backtracking Based Environments, *Annals of Operations Research* 118, 101–119, 2003.
[2] N. Beldiceanu, M. Carlsson and J.-X. Rampon, Global Constraint Catalog, SICS Technical Report T2005:08. Current version available at `http://www.emn.fr/x-info/sdemasse/gccat/`
[3] N. Beldiceanu and T. Petit, Cost Evaluation of Soft Global Constraints, CPAIOR, 80–95, 2004.
[4] C. Bessière, E. Hebrard, B. Hnich, Z. Kiziltan and T. Walsh, SLIDE: a useful special case of the CardPath constraint, ECAI 2008, 475–479, 2008.
[5] C. Bessière, G. Katsirelos, N. Narodytska, C.-G. Quimper and T. Walsh, Decompositions of All Different, Global Cardinality and Related Constraints, IJCAI 2009: 419–424.
[6] A. Borning, B. Freeman-Benson and M. Wilson, Constraint Hierarchies, *Lisp and Symbolic Computation* 5, 3, 223–270, 1992.
[7] A. Borning, M.J. Maher, A. Martindale and M. Wilson, Constraint Hierarchies and Logic Programming, ICLP, 149–164, 1989.
[8] S. Brand, N. Narodytska, C.-G. Quimper, P.J. Stuckey and T. Walsh, Encodings of the Sequence Constraint, CP 2007, LNCS 4741, Springer, 210–224.
[9] R. Dechter, *Constraint Processing*, Morgan Kaufmann, 2003.
[10] W-J. van Hoeve, G. Pesant and L-M. Rousseau, On Global Warming: Flow-Based Soft Global Constraints, *Journal of Heuristics*, 12(4-5), 347–373, 2006.
[11] W-J. van Hoeve and J-C. Régin, Open Constraints in a Closed World, CPAIOR'06, LNCS 3990, Springer, 244–257, 2006.
[12] M.J. Maher, Analysis of a Global Contiguity Constraint, Proc. Workshop on Rule-Based Constraint Reasoning and Programming, 2002.
[13] M.J. Maher, Open Contractible Global Constraints, IJCAI 2009, 578–583.
[14] M.J. Maher, Open Constraints in a Boundable World, CPAIOR 2009, LNCS 5547, 163–177.
[15] M.J. Maher, SOGgy Constraints: Soft Open Global Constraints, CP 2009, LNCS 5732, 584–591, 2009.
[16] M. Maher, N. Narodytska, C.-G. Quimper and T. Walsh, Flow-based propagators for the SEQUENCE and related global constraints, CP 2008, LNCS 5202, 159–174.
[17] T. Petit and E. Poder, The Soft Cumulative Constraint, Research Report TR09/06/info, Ecole des Mines de Nantes, 2009.
[18] T. Petit, J-C. Régin and C. Bessière, Specific Filtering Algorithms for Over-constrained Problems, CP 2001, LNCS 2239, Springer, 451–463.
[19] F. Rossi, P. van Beek and T. Walsh (Eds), *Handbook of Constraint Programming*, Elsevier, 2006.
[20] C. Schulte and G. Tack, Weakly Monotonic Propagators, CP 2009, LNCS 5732, 723–730, 2009.