

SOGgy Constraints: Soft Open Global Constraints

Michael J. Maher

NICTA* and University of NSW
Sydney, Australia
Michael.Maher@nicta.com.au

Abstract. We investigate soft open constraints. We generalize and unify classes of soft constraints and adapt them to the open setting. We give sufficient conditions for generalized classes of decomposition-based and edit-based soft constraints to be contractible. Finally, we outline a propagator for an open generalized edit-based soft REGULAR constraint.

1 Introduction

Open global constraints [1, 8, 11] allow variables to be added to the global constraint during execution, which is vital when we want to interleave problem construction and problem solving. Soft constraints are useful for addressing problems that might be overconstrained. In this paper we investigate soft open constraints, that is, soft constraints in the sense of Petit *et al* [14] that are dynamic, or open, in the sense of Barták [1]. Until now, such constraints have not been investigated.

The main focus is on the ability to recognise when soft constraints are contractible – a property that ensures that propagators for closed constraints can be re-used for the corresponding open constraint. We give sufficient conditions for generalized classes of decomposition-based and edit-based soft constraints to be contractible. We also outline a propagator for an open generalized soft REGULAR constraint to demonstrate the ease with which a propagator for a closed constraint can be adapted for an open constraint.

The reader is assumed to have a basic knowledge of constraint programming, CSPs, global constraints, and filtering, as might be found in [6, 15, 3]. In particular, global constraints not defined here can be found in [3]. Background on open constraints is given in Section 2, while Section 3 is the main section, discussing when soft constraints are contractible, and Section 4 outlines a propagator for a soft open REGULAR constraint.

2 Open Constraints

In this paper, we view a global constraint as a relation over a sequence of variables $[X_1, X_2, \dots, X_n]$ (also denoted by \mathbf{X}). Other arguments of a constraint are considered parameters and are assumed to be fixed before execution.

* 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.

We formalize the semantics of a constraint C as a formal language L_C . A word $d_1d_2\dots d_n$ appears in L_C iff the constraint $C([X_1, X_2, \dots, X_n])$ has a solution $X_1 = d_1, \dots, X_n = d_n$. 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 $P(L) = L$.

In an open global constraint, variables may be appended to the end of the sequence of variables during execution. A constraint may also be closed during execution, at which point no further variables may be appended. The scope of such constraints changes during the execution, and we refer to the state of the constraint at some point in the execution as an *occurrence* of the constraint.

There are several models of open constraint that have been proposed. Barták’s model [1] is straightforward: the constraint involves a sequence of variables to which variables may be added. Thus the type of the constraint is unchanged, whether the constraint is open or closed. The model of van Hove and Régis [8] uses a set variable S describing a set of object variables, rather than a sequence, to represent the collection of variables in the constraint. A third model [11] uses a sequence of variables, with an additional integer variable N denoting the length of the sequence. In this paper we focus on Barták’s model.

In general, a propagator for a closed constraint may be unsound for the corresponding open constraint. That is, the propagator may make an inference that turns out to be unjustified once the sequence of variables \mathbf{X} is extended to \mathbf{XY} . The property of *contractibility* was introduced in [10] to characterize those constraints for which a closed propagator is sound for the open constraint, that is, all domain reductions remain sound if the sequence of variables is extended. It was shown there that a constraint C is contractible iff L_C is prefix-closed. For contractible constraints it appears to be straightforward to extend closed implementations to open constraints [10]. Propagators also have been designed for some incontractible constraints [8, 11].

3 Contractibility of Soft Constraints

We consider “soft” constraints in the style of [14]¹. 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 $m(\mathbf{X}) \leq Z$, where m is the violation measure². We refer to the hard constraint as $C(\mathbf{X})$ and the corresponding soft constraint as $C_s(\mathbf{X}, Z)$.

Assessing the contractability of such constraints is made easier by a result of [10]. We say a function f on a sequence is *non-decreasing* if $f(\mathbf{a}) \leq f(\mathbf{b})$ for every \mathbf{a} and \mathbf{b} where \mathbf{a} is a prefix of \mathbf{b} . (Here the notation \mathbf{a} denotes a sequence $a_1 \dots a_n$ or $[a_1, \dots, a_n]$.)

¹ Consideration of other forms of soft constraint are left for future research.

² Also called violation cost [14]. We assume that violation measures return non-negative values, and return 0 exactly when the hard constraint holds.

Proposition 1. *Let C be a global constraint and suppose C can be expressed as $f(\mathbf{X}) \leq Z$. Then C is contractible iff f is a non-decreasing function.*

Thus, to evaluate whether or not soft constraints are contractible we must consider the form of the violation measure, and whether it forms a non-decreasing function.

We can recognise four broad classes of violation measures: those based on constraint decomposition, edit distance, graph properties, and the semantics of a specific constraint. We address these classes in the following subsections.

3.1 Decomposition-based Violation Measures

In the *decomposition-based* violation measures³, a decomposition of C into an equivalent conjunction of more elementary constraints is identified, and the violation measure of C_s for a particular assignment is the number of elementary constraints in the decomposition that are violated (i.e. not satisfied) by the assignment. Some work prefers the decomposition into binary constraints [14, 7], but there seems no reason to be limited by this restriction, and any canonical decomposition should serve.

When the decomposition of $C(\mathbf{X})$ is a subset of the decomposition of $C(\mathbf{XY})$ then it is clear that the number of violations for any assignment cannot decrease when the sequence of variables is extended. Hence, the violation measure is non-decreasing and the soft constraint C_s is contractible.

We can extend this result to decompositions that introduce new variables. Let $C(\mathbf{X}) \leftrightarrow \exists \mathbf{U} \bigwedge_{c \in \mathcal{C}} c(\mathbf{X}, \mathbf{U})$ be the decomposition of $C(\mathbf{X})$, where \mathcal{C} is the collection of elementary constraints in the decomposition of $C(\mathbf{X})$. We define the violation measure m as follows: for every assignment v to the variables \mathbf{X} , $m(\mathbf{X}) = \min \text{num_viol}_{v'}(\mathcal{C})$ where we minimize over all extensions v' of v to \mathbf{U} , and $\text{num_viol}_{v'}(S)$ is the number of unsatisfied constraints in S under v' .

We assume that the decomposition can be expressed as a formula of the form $\exists \mathbf{U} \bigwedge_i p_i(\mathbf{X}, \mathbf{U})$, where each $p_i(\mathbf{X}, \mathbf{U})$ denotes an occurrence of an elementary constraint over a subset of the variables \mathbf{X}, \mathbf{U} , and constants. We permit duplicate occurrences of a constraint, which allows a form of weighting within a decomposition. We denote such a formula by $(\mathbf{X}, \mathbf{U}, S)$, where S is the collection of elementary constraints. We say that one formula $(\mathbf{X}, \mathbf{U}, S)$ is *covered* by another formula $(\mathbf{W}, \mathbf{V}, T)$ if there is a variable renaming θ that maps \mathbf{X} into \mathbf{W} and \mathbf{U} into $\mathbf{V} \cup \mathbf{W}$ such that the multiset $S\theta$ is a submultiset of T ⁴.

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

Proposition 2. *Let C_s be a soft constraint with a decomposition-based violation measure. Let $(\mathbf{X}, \mathbf{U}, S)$ be the decomposition of $C(\mathbf{X})$ and $(\mathbf{XY}, \mathbf{V}, T)$ be the decomposition of $C(\mathbf{XY})$. If $(\mathbf{X}, \mathbf{U}, S)$ is covered by $(\mathbf{XY}, \mathbf{V}, T)$ via a renaming that is the identity on \mathbf{X} then C_s is contractible.*

³ This includes primal graph based violation costs [14].

⁴ A multiset M_1 is a *submultiset* of multiset M_2 if, for every element $a \in M_1$, the multiplicity of a in M_1 is less than or equal to its multiplicity in M_2 .

For example, CONTIGUITY is implemented in [9] essentially by the decomposition

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

for a certain constraint C' . The decomposition of CONTIGUITY(\mathbf{XY}) covers that of CONTIGUITY(\mathbf{X}), and hence a soft CONTIGUITY constraint under the decomposition-based violation measure is contractible.

In general, any constraint definition based on composition of smaller constraints can be viewed as a decomposition. Thus, as a consequence of the previous Proposition, we have a counterpart of results of [10] for decomposition-based soft constraints.

Corollary 1. *If C is a global constraint definable using only the meta-constraints SLIDE and SPLASH, existential quantification, conjunction, and constraints on a fixed finite prefix of \mathbf{X} , then the soft constraint C_s based on that decomposition is contractible.*

As a result, decomposition-based soft ALLDIFFERENT, INTERDISTANCE, SEQUENCE and \leq_{lex} constraints are contractible, using the decompositions mentioned in [10]. Different decompositions for the same constraint may lead to different violation measures, with possibly different contractibility of the corresponding soft constraints. Thus it is necessary to specify the decomposition involved.

Rather than using $num_viol_{v'}$ to count the *number* of violations in a decomposition, we can instead sum the *amount* of violation (by v') of the constraints in the decomposition. This gives rise to the value-based violation measure for GCC [14, 7] and the measure used for the soft SEQUENCE constraint [12]. In the latter case, the decomposition of $C(\mathbf{X})$ is a subset of the decomposition of $C(\mathbf{XY})$ and it follows that the soft SEQUENCE constraint is contractible. More generally

Proposition 3. *If $C(\mathbf{X})$ is covered by $C(\mathbf{XY})$ via a substitution θ that leaves \mathbf{X} unchanged then C_s is contractible under such measures.*

This proposition does not apply to the value-based soft GCC because the decomposition of $C(\mathbf{X})$ involves constraints over \mathbf{X} . These constraints $c(\mathbf{X})$ cannot be covered by the corresponding $c(\mathbf{XY})$. In fact, the value-based soft GCC is not contractible, for essentially the same reason that the hard GCC is not contractible: lower bounds on the number of occurrences of a value can lead to the violation measure decreasing as the sequence gets longer.

Covering is a purely syntactic relationship; it is independent of the meaning of the constraints involved. Semantically equivalent (conjunctions of) constraints do not, in general, cover each other. To take a trivial example, constraints $X+0 = Y$ and $X*1 = Y$ are semantically equivalent, but do not cover each other because they are syntactically different. As a result, Propositions 2 and 3 may fail to recognise cases of contractibility.

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. This class includes the *variable-based* violation measures [14, 7], since such measures are simply edit distances where substitution is the only edit operation. The *object-based* measures of [4] are edit distances where deletion is the only edit operation.

The edit-based violation measures for closed constraints are not appropriate for open constraints, because they fail to take into account that the current sequence of variables may be extended with more variables.

For example, consider an open constraint C where $L_C = abc + defghi$ and an occurrence of the constraint $C([X_1, X_2, X_3])$. If $X_1 = d$, $X_2 = e$ and $X_3 = f$ then the edit distance of this instance to L_C is 3, even though this instance is completely accurate if the sequence of variables is extended. Similarly, if $L_C = abc$ and we have an occurrence $C([X_1, X_2])$ with $X_1 = a$ and $X_2 = b$ then the edit distance is 1, even though there is no violation.

To take account of the possibility that a sequence of variables may be extended, we employ the edit distance to $P(L_C)$, the prefix-closure of L_C . With this choice, the violation measure in any occurrence is a lower bound on the violation measure of the final occurrence of the constraint.

In [10] the prefix-closure was used to approximate a constraint so that constraint propagation is sound when the constraint is open. In [11] the approximation was used to design propagators for specific constraints. The use of the prefix closure in this paper is somewhat different from its use in [10, 11]: rather than using $P(L_C)$ as an approximation to L_C , $P(L_C)$ is used to formulate what it means to be an open soft constraint.

To address a wide range of edit-based measures, we generalize the measures. We allow non-negative weights $\alpha, \beta, \gamma, \delta$ 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 we define $m(w) = \min_{\text{edits}} \alpha n_s + \beta n_i + \gamma n_d + \delta n_t$ to be the minimum, over all edits that transform w to an element of $P(L_C)$, of the weighted sum of the edit operations.

Proposition 4. *Let C_s be a soft constraint with a weighted edit-based violation measure. Suppose $\alpha \leq \delta$ or $\beta \leq \delta$.*

Then C_s is contractible.

It follows that edit-based measures that only involve substitutions, insertions and deletions provide contractible constraints. Thus the variable-based measures [14, 7], the object-based measures [4], and the edit-based measures of [7] induce contractible soft constraints. However, when transpositions are allowed and have a comparatively low cost, an edit-based violation measure can lead to a soft constraint that is not contractible.

Example 1. Suppose $\delta < \alpha$, $\delta < \beta$, and $\delta < \gamma$. Consider a REGULAR constraint for $(ab)^* + (ab)^*a$, which is a prefix-closed language. The word *abba* has edit distance δ , by transposing the last two letters, but its prefix *abb* has edit distance $\min\{\alpha, \beta, \gamma\}$, since we could either substitute *a* for *b*, insert *a* before the second *b*, or delete a *b*. Thus the weighted edit-distance violation measure is not non-decreasing and hence, by Proposition 1, the corresponding soft open REGULAR constraint is not contractible.

This example reinforces a point made earlier: the introduction of $P(L_C)$ to the definition of edit-based violation measure plays a different role than its use in [10, 11]; in this case, its use does not ensure contractibility.

3.3 Graph Property-based Violation Measures

The *graph property-based* violation measures [4] are based on representation of global constraints by graph properties [2, 3]. A global constraint C is represented by an initial directed graph G_i , where each vertex corresponds to a variable and each edge has an attached constraint on the variables, and a graph property⁵. An assignment v to variables is a solution to C iff the graph G_f , obtained by deleting all edges whose constraints are not solved by v and all isolated vertices, satisfies the graph property. A graph property is a simple constraint on a characteristic of G_f such as the number of vertices, edges, sources or sinks, or the number or size of connected components or strongly connected components, etc. The violation measure of C under v is the violation measure of the graph property on G_f .

This framework has great flexibility; [4] presents three different violation measures for ALLDIFFERENT in this framework. Unfortunately, this makes it difficult to identify classes of constraints that are contractible. We need to narrow our focus to get a result.

Proposition 5. *Let C be a global constraint, defined in the graph property framework, and C_s be the corresponding graph property-based soft constraint.*

Suppose $G_i^{\mathbf{X}}$ is a subgraph of $G_i^{\mathbf{XY}}$ for every sequence \mathbf{X} ⁶; the only graph characteristics used to define C are the number of vertices, number of arcs, and the size of the largest (strongly) connected component in G_f ; and the graph properties have the form of upper bounds on these graph characteristics.

Then C_s is contractible.

This result is applicable to CONTIGUITY as defined in [3] and the third form of ALLDIFFERENT in [4] (slightly altered to use \leq in place of $=$ in the graph property, which is necessary in an open setting). It also applies to ATMOST, DISJOINT, and GOLOMB.

The assumption that $G_i^{\mathbf{X}}$ is a subgraph of $G_i^{\mathbf{XY}}$ is not a strong assumption, given that most arc generators in [3] have this property. However, the other assumptions are very limiting.

⁵ In general, multiple graphs and/or graph properties may be used.

⁶ We use the sequence of variables as a superscript, to distinguish graphs.

3.4 Other Violation Measures

The last class of violation measures are *ad hoc* measures derived from the semantics of C . Because formulations of the measures are related to the semantics of the specific constraint and, perhaps, to a specific application, it is difficult to make broad statements about the nature of these measures and, hence, the contractibility of the corresponding soft constraints.

4 Propagator

The focus of this paper is not propagators, but we give a brief outline of a propagator for an open soft REGULAR under a weighted edit-based measure, building on the minimum cost network flow propagator designed in [7]. As discussed in Section 3.2, we use the prefix-closure of the given automaton. The propagator of [7] addressed only substitutions, insertions and deletions, but it can be extended to also handle transpositions. Transposition violation arcs have the form (q_k^i, q_l^{i+2}) where $\delta(q_k, d_{i+1}) = q$ and $\delta(q, d_i) = q_l$ for some state q and some $d_i \in D(X_i)$, $d_{i+1} \in D(X_{i+1})$. Violation arcs of the different types have capacity 1 and cost α, β, γ or δ , depending on type. When a new variable is appended to the sequence of variables, an additional copy of the states of the automaton must be added, with additional arcs representing both automata transitions and violations, somewhat like the open REGULAR constraint in [11]. The propagation algorithm is essentially the same as the one for edit-based SOFTREGULAR in [7]. The only difference is that reductions in variable domains may require transposition violation arcs to be deleted.

Because some of the variables in an open constraint will be unspecified during part of the execution, we need to adapt the definition of consistency. The following is an appropriate form of domain consistency for Barták's model [11].

Definition 1. *Given a domain D , an occurrence of a constraint $C(\mathbf{X})$ is open D -consistent if*

for every $X_i \in \mathbf{X}$ and every $d \in D(X_i)$ there is a word $d_1 \dots d_n$ in L_C such that $d_i = d$, $|\mathbf{X}| \leq n$, and $d_j \in D(X_j)$ for $j = 1, \dots, |\mathbf{X}|$.

Proposition 6. *Suppose $\alpha \leq \delta$ or $\beta \leq \delta$, and $\beta + \gamma \leq 2\delta$.*

The propagator described above achieves open D -consistency for the soft open global REGULAR constraint under the weighted edit-distance violation measure.

The proof relies on a lemma that states that if $\beta + \gamma \leq 2\delta$ then there is an edit of minimal cost where any letter subject to transposition is not subject to any other edit operation. It also relies on Proposition 4 above, Proposition 8 of [11] and Corollary 6 of [7].

5 Conclusions

We have broadened some classes of violation measures, and used these classes to establish that a wide range of soft constraints are contractible. Notice that

the contractibility of a soft constraint is independent of the contractibility of the underlying hard constraint. The edit-based soft REGULAR constraint is contractible if all edit operations have weight 1, even when the underlying REGULAR constraint is not contractible. Conversely, a contractible hard constraint can give rise to an uncontractible soft constraint if the violation measure is chosen badly, as we saw in Example 1. Indeed, it appears that contractibility depends more on the violation measure than on the hard constraint. For example, soft GCC is contractible under many edit-based measures, such as the variable-based measure, but not under the decomposition-oriented value-based measure.

Contractibility only supports the adaptation of closed constraint propagators to open propagators. It seems orthogonal to issues like the usefulness of the violation measure for an application, the complexity of propagation, and the design of efficient propagators. These issues require further research.

Acknowledgements Thanks to the referees for their insightful comments.

References

1. R. Barták, Dynamic Global Constraints in Backtracking Based Environments, *Annals of Operations Research* 118, 101–119, 2003.
2. N. Beldiceanu, Global Constraints as Graph Properties on a Structured Network of Elementary Constraints of the Same Type, CP 2000, LNCS 1894, Springer, 52–66.
3. 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/>
4. N. Beldiceanu and T. Petit, Cost Evaluation of Soft Global Constraints, CPAIOR, 80–95, 2004.
5. C. Bessière, E. Hebrard, B. Hnich, Z. Kiziltan and T. Walsh, SLIDE: a useful special case of the CardPath constraint, ECAI 2008, 475–479.
6. R. Dechter, *Constraint Processing*, Morgan Kaufmann, 2003.
7. W.-J. van Hoesve, G. Pesant and L.-M. Rousseau, On Global Warming: Flow-Based Soft Global Constraints, *Journal of Heuristics*, 12(4-5), 347–373, 2006.
8. W.-J. van Hoesve and J.-C. Régin, Open Constraints in a Closed World, CP-AI-OR’06, LNCS 3990, Springer, 244–257, 2006.
9. M.J. Maher, Analysis of a Global Contiguity Constraint, Proc. Workshop on Rule-Based Constraint Reasoning and Programming, 2002.
10. M.J. Maher, Open Contractible Global Constraints, IJCAI 2009, to appear.
11. M.J. Maher, Open Constraints in a Boundable World, CPAIOR 2009, LNCS 5547, 163–177.
12. 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.
13. G. Pesant, A Regular Language Membership Constraint for Finite Sequences of Variables. CP 2004, LNCS 3258, Springer, 482–495.
14. T. Petit, J.-C. Régin and C. Bessière, Specific filtering algorithms for over-constrained problems, CP 2001, LNCS 2239, Springer, 451–463.
15. F. Rossi, P. van Beek and T. Walsh (Eds), *Handbook of Constraint Programming*, Elsevier, 2006.