Towards Efficient Consistency Enforcement for Global Constraints in Weighted Constraint Satisfaction

J. H. M. Lee and K. L. Leung
Department of Computer Science and Engineering
The Chinese University of Hong Kong, Shatin, N.T., Hong Kong
{jlee,klleung}@cse.cuhk.edu.hk

Abstract

Powerful consistency techniques, such as AC* and FDAC*, have been developed for Weighted Constraint Satisfaction Problems (WCSPs) to reduce the space in solution search, but are restricted to only unary and binary constraints. On the other hand, van Hoeve et al. developed efficient graph-based algorithms for handling soft constraints as classical constraint optimization problems. We prove that naively incorporating van Hoeve’s method into the WCSP framework can enforce a strong form of \(\emptyset\)-Inverse Consistency, which can prune infeasible values and deduce good lower bound estimates. We further show how Van Hoeve’s method can be modified so as to handle cost projection and extension to maintain the stronger AC* and FDAC* generalized for non-binary constraints. Using the soft allDifferent constraint as a testbed, preliminary results demonstrate that our proposal gives improvements up to an order of magnitude both in terms of time and pruning.

1 Introduction

The task at hand is how to relax or weaken some of the hard constraints in an over-constrained problem so as to obtain useful partial solutions. Weighted constraint satisfaction [Schiex et al., 1995] is a framework for handling such tasks. While the basic technique for solving weighted constraint satisfaction problems (WCSPs) relies on a form of branch-and-bound search, various consistency notions and techniques [Larrosa and Schiex, 2003; 2004; Sanchez et al., 2008] for unary, binary, and ternary constraints have been developed to help prune the search space. Higher arity constraints have to be either first converted to their binary counterparts or activated only after enough variables are instantiated during search. The lack of efficient handling of non-binary constraints in WCSP systems greatly restricts the applicability of WCSP techniques to complex real-life problems.

Incorporating arbitrary soft \(n\)-ary constraints into WCSP can be difficult since the costs have to be represented extensionally and maintained in an \(n\)-dimensional table, incurring time and space overheads. Soft global constraints are non-binary constraints with semantics. In particular, the cost structure of flow-based soft global constraints [van Hoeve et al., 2006] can be formulated as a flow network, allowing the computation of the minimum cost of the soft global constraints using minimum cost flow algorithm. This is useful in estimating the lower bound of the current search path. We show that a naive incorporation of flow-based soft global constraints into WCSP would result in a strong form of the \(\emptyset\)-inverse consistency [Zytnicki et al., 2006], which is still relatively weak in terms of lower bound estimation and pruning. The question becomes whether we can achieve stronger consistencies, the generalized versions of AC* [Larrosa and Schiex, 2004] and FDAC* [Larrosa and Schiex, 2003], for non-binary constraints efficiently. Consistency algorithms for AC* and FDAC* involve three main operations: (a) computing the minimum cost of the constraint when a variable \(x\) is fixed with value \(v\), (b) projecting the minimum cost of a constraint to the unary constraint for \(x\) at value \(v\), and (c) extending the unary cost to the non-unary constraints. These operations allow cost movement among constraints and shifting of cost to the \(C_\emptyset\) constraint, resulting in higher lower bound and also domain prunings. Part (a) is readily handled by the minimum cost flow (MCF) algorithm. We show how the MCF algorithm and the corresponding flow networks can be adapted for parts (b) and (c) so as to perform projection and extension in polynomial time and space complexity. Using the soft allDifferent constraint as a testbed, we demonstrate the advantages of the stronger consistencies over naive incorporation.

2 WCSP

The weighted CSP (WCSP) framework extends classical constraint satisfaction by associating costs to the tuples of variable assignments. A WCSP [Schiex et al., 1995] is a tuple \((X, D, C, k)\). \(X\) is a set of variables \(\{x_1, x_2, \ldots, x_n\}\) ordered by their indices. \(D\) is a set of domains \(D(x_i)\) for \(x_i \in X\). Each \(x_i\) can only be assigned one value in its corresponding domain. An assignment on a set of variables can

\*We thank the anonymous referees for their constructive comments. The work described in this paper was substantially supported by grants CUHK413207 and CUHK413808 from the Research Grants Council of Hong Kong SAR.
be represented by a tuple \( \ell \). We denote \( \ell[x_i] \) as the value assigned to \( x_i \), and \( \ell[S] \) as the tuple formed from the assignment on a subset of variables \( S \). \( C \) is a set of constraints, each \( C_S \) of which represents a function mapping tuples corresponding to assignments on \( S \) to a cost valuation structure \( V(k) = \{0, \ldots, k\} \). The structure \( V(k) \) contains a set of integers \([0 \ldots k] \) with standard integer ordering \( \leq \). Addition \( \oplus \) is defined by \( a \oplus b = \text{min}(k, a + b) \), and subtraction \( \ominus \) is defined by \( a \ominus b = a - b \) if \( a \neq k \) and \( a \ominus a = k \) for any \( a \).

Without loss of generality, \( C_S \) can always be defined (initially with all tuples mapping to zero) for all \( S \subseteq X \). The cost of a tuple \( \ell \) corresponding to an assignment on \( X \) is defined as:

\[
\text{cost}(\ell) = C_\emptyset \oplus \bigoplus_{C_S \in C} C_S(\ell[S]),
\]

where \( C_\emptyset \) is a null constraint that denotes the lower bound of costs of all possible tuples. A tuple \( \ell \) corresponding to an assignment on \( X \) is feasible if \( \text{cost}(\ell) < k \), and is a solution of a WCSP if \( \ell \) has the minimum cost among all feasible tuples.

A soft global constraint \( C_S \) on variables \( S \) has a particular semantics and can have more than one cost measure. Where necessary, we sometimes give also a separate cost function \( \mu \) in case \( C_S \) has more than one such function. For simplicity, we assume that when we write \( C_S(\ell) \) or \( \mu(\ell) \), \( \ell \) is always a feasible assignment to \( S \).

WCSPs are solved with basic branch-and-bound search augmented with consistency techniques which prune infeasible values from variable domains and push lower bound estimates into \( C_\emptyset \). Common consistency notions and techniques [Larrosa and Schiex, 2003; 2004; Sanchez et al., 2008] include NC*, AC*, and FDAC*, but are designed for unary to ternary constraints only.

A variable \( x_i \) is NC* if (1) each value \( v \in D(x_i) \) satisfies \( C_{x_i}(v) \oplus C_\emptyset \leq k \) and (2) there exists a value \( v' \in D(x_i) \) such that \( C_{x_i}(v') = 0 \). A WCSP is NC* iff all variables are NC*.

Algorithm 1 enforces NC*. Function unaryProject() projects costs from unary constraints to \( C_\emptyset \) by simple arithmetic operations, and pruneVal() removes infeasible values from domains.

### 3 Enforcing \( \emptyset \)IC and Strong \( \emptyset \)IC

In this section, we explain how van Hoeve’s method of using minimum cost flow can be adapted for WCSPs to enforce a strong form of \( \emptyset \)-inverse consistency [Zytnicki et al., 2006]. Given a connected flow network \( G(V, E, w, c, s, t) \), where \( V \) are the vertices, \( E \) are the edges, and each edge \( e \in E \) has a weight \( w_e \) and a capacity \( c_e \). A flow \( f \) from a source \( s \) to a sink \( t \) of a value \( \alpha \) in \( G \) is defined as a mapping from \( E \) to \( \mathbb{R} \) such that:

- \( \sum_{(u,s) \in E} f_{su} = \sum_{(u,t) \in E} f_{ut} = \alpha \);
- \( \sum_{(u,s) \in E} f_{wu} = \sum_{(u,t) \in E} f_{wu} \forall v \in V \setminus \{s, t\} \);
- \( 0 \leq f_e \leq c_e \forall e \in E \).

If \( \alpha \) is not defined, \( \alpha \) is the maximum value of all flows in \( G \). The cost of a flow \( f \) is defined as \( \sum_{e \in E} w_e f_e \). A soft global constraint \( C_S \) with cost function \( \mu \) is flow-based if \( \mu \) allows for a representation in a flow network \( G \) so that the flow with minimum cost in \( G \) corresponds to the tuple mapping to the minimum cost in \( C_S \). Van Hoeve et al. [2006] demonstrate his framework on the soft versions of the allDifferent, gcc, regular, and same constraints.

We use the soft allDifferent constraint with the \( \mu_{\text{dec}} \) cost measure [Petit et al., 2001] to illustrate the concepts. Given an assignment tuple \( \ell \) on variables \( S \), \( \mu_{\text{dec}}(\ell) = |\{(i,j) | i < j \land \ell[x_i] = \ell[x_j] \land x_i, x_j \in S\}| \), which stands for the number of pairs of variables sharing the same value. We can construct a flow network \( G(V, E, w, c, s, t) \) as follows [van Hoeve et al., 2006]. The network consists of \( |S| + |D| + 2 \) nodes, where \( |D| \) is the size of the union of all variable domains in \( S \). Each variable and value have an associated node, with two more nodes \( s \) and \( t \). The network contains three sets of edges:

- \( (s, x_i) \in E \) for each \( x_i \in X \) with zero weight and unit capacity;
- \( (x_i, v) \in E \) for each \( v \in D(x_i) \) with zero weight and unit capacity;
- \( (v, t) \in E \), for each \( i = 1, \ldots, d_v \), where \( d_v \) is the number of values containing \( v \). Each edge \( (v, t) \) has a unit capacity and a weight of \( i - 1 \).

For example, if \( X = \{x_1, x_2, x_3, x_4\} \) with \( D(x_1) = \{a, c\} \), \( D(x_2) = \{b, d\} \), \( D(x_3) = \{a, d\} \) and allDifferent \( (X) \), the network is constructed as shown in Figure 1. Only non-zero weights are shown in the network. All edges assume unit capacity. The minimum cost of the feasible flow in \( G \) with value \( |X| \) is \( \min\{\mu_{\text{dec}}(\ell)\} \). To compute \( \min\{\mu_{\text{dec}}(\ell) \mid \ell[x_i] = v \} \), the minimum cost flow simply enforces \( f_{sv} = 1 \). For example, Figure 1 shows a flow (highlighted by thickened edges) of minimum cost when \( f_{x_1 a} = 1 \). Regin [2002] and Van Hoeve et al. [2006] proved that such an enforcement can be derived from an existing flow.
by constructing a residual network from \( G \) and the existing flow, and finding the minimum cost cycle containing \( (x, v) \) in the residual network. This can be found by using the single source shortest path algorithm.

We now define \( \varnothing IC \) [Zytynicki et al., 2006] and strong \( \varnothing IC \) for WCSPs, the enforcement of which can benefit from van Hoeve’s method. A constraint \( C_S \) is \( \varnothing IC \) if there exists a tuple \( \ell \) corresponding to a feasible assignment with \( C_S(\ell) = 0 \). A WCSP is \( \varnothing IC \) iff all constraints are \( \varnothing IC \).

For example, Figure 2(a) shows a WCSP which is not \( \varnothing IC \). No matter which values are assigned to the variables, \( C_{x_1, x_2} \) returns a cost of at least 1. To enforce \( \varnothing IC \), a cost of 1 is projected directly from \( C_{x_1, x_2} \) to \( C_{\varnothing} \) by reducing the cost of each tuple by 1 and increase \( C_{\varnothing} \) by 1. The resultant WCSP is shown in Figure 2(b).

Procedure \( \text{enforce}\varnothing IC() \) in Algorithm 2 enforces \( \varnothing IC \) for a WCSP by enforcing \( \varnothing IC \) on each constraint. Reducing the cost of each tuple can be expensive. An implementation trick is to use a zero-initialized variable \( z_S \) to store the cost reduced so far due to projection from \( C_S \) to \( C_{\varnothing} \). If a tuple \( \ell \) queries its cost from \( C_S \), the result is \( C_S(\ell) \oplus z_S \).

In general, the algorithm is exponential even with the implementation trick since exponential number of tuples have to be examined at line 5. However, minimum cost flow computation allows for a polynomial time algorithm for flow-based soft global constraints, such as allDifferent with \( \mu_{\text{dec}} \).

Enforcing \( \varnothing IC \) only increases \( C_{\varnothing} \). We observe, for example in Figure 2(b), that the value \( d \in D(x_1) \) cannot be part of any solution. The tuple associated with \( x_1 = d \) has a cost at least 4: 1 from \( C_{\varnothing} \), 2 from \( C_{x_1} \), and 1 from \( C_{x_1, x_2} \). Extra conditions can be added to strengthen \( \varnothing IC \) to allow also domain reduction. A non-unary constraint \( C_S \) is strong \( \varnothing IC \) if:

- \( C_S \) is \( \varnothing IC \), and;
- for all values \( v \in D(x) \) with \( x \in S \), \( C_{\varnothing} \oplus C_x(v) \oplus \min\{C_S(\ell) | t[x] = v\} < k \).

A WCSP is strong \( \varnothing IC \) iff all constraints are strong \( \varnothing IC \).

For example, the WCSP in Figure 2(b) is not strong \( \varnothing IC \), but removing the value \( d \) from \( D(x_1) \) makes it so. Procedure \( \text{enforceStrong}\varnothing IC() \) in Algorithm 3 enforces strong \( \varnothing IC \), based on the \( \wedge\text{-AC*3()} \) Algorithm [Larrosa and Schiex, 2004]. The algorithm maintains a propagation queue \( Q \) (implemented as a set) of variables. Constraints involving variables in \( Q \) are potentially not strong \( \varnothing IC \). Function \( \text{pop()} \) removes an arbitrary available variable from \( Q \) in constant time.

![Figure 1: A flow network for allDifferent. The thick edges give the minimum cost flow when \( x_1 = a \).](image)

![Figure 2: Two equivalent WCSPs with \( k = 4 \).](image)
Procedure \texttt{enforceStrongG\textsubscript{IC}()} in Algorithm 3 must terminate, the proof of which is similar to those of Larrosa and Schiex’s Theorems 12 and 21 [2004]. Suppose \texttt{removeInfeasible()} and \texttt{enforceG\textsubscript{IC}()} have a time complexity of $O(f_{\text{strong}})$ and $O(f_{\emptyset})$ respectively, the complexity can be stated as follows.

**Theorem 1** Procedure \texttt{enforceStrongG\textsubscript{IC}()} has a time complexity of $O(\max\{d \cdot f_{\text{strong}} + f_{\emptyset}\})$, where $e$ is the number of non-unary constraints, $s_{\max}$ is the maximum arity of the constraints, $n$ is the number of variables, and $d$ is the maximum domain size. Thus, \texttt{enforceStrongG\textsubscript{IC}()} must terminate.

**Proof:** In each iteration of the while loop, line 6 will be executed $O(s_{\max}e)$ times. Each variable is pushed into $Q$ at most $O(d)$ times due to line 7 (each time $D(x_u)$ is modified); thus the while loop will be executed $O(s_{\max}d)$ times. Therefore, the complexity of procedure \texttt{enforceStrongG\textsubscript{IC}()} is $O(s_{\max}d(s_{\max}e f_{\text{strong}} + f_{\emptyset}))$, and it must terminate.

Again, \texttt{enforceStrongG\textsubscript{IC}()} requires exponential complexity since line 12 is exponential in general. However, line 12 can be computed in polynomial time for flow-based soft global constraints. The following result is a consequence of Regin’s Lemma 1 [2002] and van Hoeve et al.’s Theorem 1 [2006].

**Theorem 2** If $C_S$ is a flow-based soft global constraint, \texttt{removeInfeasible()} has a time complexity of $O(K + d \cdot SP)$, where $O(K)$ and $O(SP)$ are the time complexity to find the minimum cost flow and single source shortest path respectively, and $d$ is the maximum domain size.

Given a network $G(V, E, w, c, s, t)$. A typical $O(K)$ is $O(|V|^2|E|)$ if the successive shortest path algorithm is used, and a typical $O(SP)$ is $O(|V||E|)$ if a label correcting algorithm, like the Bellman-Ford algorithm, is used [Ahuja et al., 2005].

Due to space limitation, we cannot give the details of the reasoning that the “Soft as Hard” approach [Petit et al., 2001] is slightly weaker than enforcing strong $\emptyset$IC together with NC*, which is still relatively weak in terms of the deduced lower bound and pruning. Stronger consistencies for soft global constraints are desirable.

## 4 Projection in GAC*

We specialize the definition of GAC in Cooper et al. [2004] for WCSP. A variable $x_i \in S$ is generalized arc consistent star (GAC*) with respect to a non-unary constraint $C_S$ if:

- $x_i$ is NC*, and;
- for each value $v_i \in D(x_i)$, there exists values $v_j \in D(x_j)$ for all $j \neq i$ and $x_j \in S$ so that they form a tuple $t$ with $C_S(t) = 0$. $\{v_j\}$ is a simple support of $v_i$ with respect to $C_S$.

A WCSP is GAC* iff all variables are GAC* with respect to all constraints. Notice that GAC* collapses to AC* for binary constraints [Larrosa and Schiex, 2004] and AC for ternary constraints [Sanchez et al., 2008].

Procedure \texttt{enforceGAC*()} in Algorithm 4 enforces GAC* for a WCSP and is based on the $W\text{-AC*3()}$ Algorithm [Larrosa and Schiex, 2004]. Algorithm 4 must terminate, the proof of which is similar to that of Theorem 1. By replacing $O(f_{\text{strong}})$ and $O(f_{\emptyset})$ by $O(f_{GAC})$ (the complexity of \texttt{findSupport()} and $O(nd)$ ($n$ times the complexity of \texttt{pruneVal()} respectively), the complexity of Algorithm 4 can be stated as follows.

**Theorem 3** Procedure \texttt{enforceGAC*()} has a time complexity of $O(s_{\max}d(s_{\max}e f_{GAC} + nd))$, where $n$, $d$, $e$, and $s_{\max}$ are as defined in Theorem 1. Thus, \texttt{enforceGAC*()} must terminate.

Again, Algorithm 4 requires exponential time complexity since function \texttt{findSupport()} is exponential. The time complexity of \texttt{findSupport()} is determined by two operations: minimum cost computation (line 14) and cost projection (lines 16 to 18). Line 14 computes the minimum cost of $C_S$ when $x_i = v$. Line 16 projects the cost to the unary constraint $C_{x_i}$, which is a simple arithmetic operation. Lines 17 and 18 update the cost of all tuples corresponding to $x_i = v$. In general, this two sub-procedures require exponential time complexity, which can be reduced for flow-based soft global constraints. Van Hoeve’s method can be applied similarly to line 14 as in Section 3. Lines 17 to 18 modify the cost function of the soft (global) constraint $C_S$. Before we give our method, we state the conditions under which our method is applicable.

A soft global constraint $C_S$ with cost function $\mu$ is projection-safe if:

- the soft global constraint $C_S$ with cost function $\mu$ is flow-based, and has the corresponding flow network.
there is a one-one correspondence between every flow $f$ of $G$ and a complete variable assignment tuple $f$ for $C_S$, and

- there exists an injection from an assignment $x_i = v$ to $\bar{e} \in E$ such that whenever $\ell(x_i) = v$ for some tuple $\ell$, $f_{\bar{e}} = 1$ in the flow $f$ corresponding to $\ell$; whenever $\ell(x_i) \neq v$, $f_{\bar{e}} = 0$.

Given a projection-safe soft global constraint $C_S$ with cost function $\mu$ defined above. Suppose a cost of $\alpha$ is projected from $C_S$ to $C_{x_i}$ associated with $x_i = v$, resulting in a new cost function $\mu'$. In other words, $\mu'(\ell) = \mu(\ell) \oplus \alpha$ if $\ell(x_i) = v$; otherwise $\mu'(\ell) = \mu(\ell)$. We construct the corresponding flow network of $C_S$ with cost function $\mu'$ as $G'(V,E,w',c,s,t)$, where $w'_e = w_e \oplus \alpha$ if $e$ is the edge corresponding to $x_i = v$; otherwise $w'_e = w_e$.

We use again the allDifferent constraint with $\mu_{\text{dec}}$ as an example. Figure 3 shows the corresponding flow network and the flow representing $(x_1,x_2,x_3,x_4) = (a,b,a,d)$ with cost 1. If a cost of 1 is projected from the constraint to $C_{x_1}$ associated with $x_1 = a$, a new network can be constructed by decreasing the weight $w_{x_1}$ of the edge $(x_1,a)$ from 0 to $-1$, as shown in Figure 3. The new cost of the flow in the network is now 0, which corresponds to the cost of the tuple $(a,b,a,d)$ after projection.

Figure 3: The flow network corresponding to allDifferent after projection.

The soundness and closure of our method are guaranteed by the following theorem.

**Theorem 4** Suppose $C_S$ is a soft global constraint with cost function $\mu$ is projection safe, a cost of $\alpha$ associated with $x_i = v$ is projected from $C_S$ to $C_{x_i}$, resulting in a new cost function $\mu'$.

- (Soundness) If $f$ is a minimum cost flow of $G'(V,E,w',c,s,t)$, then $\sum_{\bar{e} \in E} w'_{\bar{e}} f_{\bar{e}} = \min \{ \mu'(\ell) \}$.
- (Closure) $C_S$ with cost function $\mu'$ is projection safe.

**Proof:** Projection-safety implies that $\sum_{\bar{e} \in E} w'_{\bar{e}} f_{\bar{e}} = \sum_{\bar{e} \in E} w_{\bar{e}} f_{\bar{e}} \oplus \alpha f_{\bar{e}} = \min \{ \mu(\ell) \} \oplus \alpha f_{\bar{e}} = \min \{ \mu'(\ell) \}$, where $\bar{e}$ is the edge corresponding to $x_i = v$. This concludes soundness.

In addition, $C_S$ with $\mu'$ is flow-based with $G'(V,E,w',c,s,t)$ as the corresponding flow network. Since the topology of $G'(V,E,w',c,s,t)$ is the same as that of $G(V,E,w,c,s,t)$, $C_S$ with $\mu'$ is projection-safe.

We state without proof that the majority of the flow-based global constraints [van Hoeve et al., 2006] are projection-safe so that GAC* can be enforced on them in polynomial time.

**Theorem 5** The following flow-based soft global constraints are projection-safe.

- allDifferent with either $\mu_{\text{var}}$ or $\mu_{\text{dec}}$;
- gcc with either $\mu_{\text{var}}$ or $\mu_{\text{val}}$;
- same with $\mu_{\text{var}}$.

Unfortunately, the regular constraint with either $\mu_{\text{var}}$ or $\mu_{\text{dec}}$ [van Hoeve et al., 2006] and the soft $\text{SEQUENCE}$ constraint [Maher et al., 2008] are not projection-safe since they do not satisfy the third requirement.

Again, the complexity of enforcing GAC* for a variable with respect to the projection-safe soft global constraints follows from van Hoeve et al.’s Theorem 1.

**Theorem 6** If $C_S$ is projection safe, $\text{findSupport}()$ has a time complexity of $O(K + d \cdot SP)$, where $K$ and $SP$ are as defined in Theorem 2.

Based on FDAC* [Larrosa and Schiex, 2003], even stronger consistency can be defined but its enforcement involves an extension operator, which is the reverse of projection and the focus of the next section.

## 5 Extension in FDGAC*

Suppose variables are ordered by their indices. A variable $x_i \in S$ is directional generalized arc consistent star (DGAC*) with respect to a non-unary constraint $C_S$ if:

- $x_i$ is NC*, and;
- for each value $v_i \in D(x_i)$, there exists values $v_j \in D(x_j)$ for all $j \neq i$ and $x_j \in S$ so that they form a tuple $\ell$ with $C_S(\ell) \oplus \bigoplus_{j \in S, x_j \in \ell} C_{x_j}(v_j) = 0$, \{v_j\} is a full support of $v_i$ with respect to $C_S$.

A WCSP is full directional generalized arc consistent star (FDGAC*) if all variables are DGAC* and GAC* with respect to all non-unary constraints. When the constraints are binary, FDGAC* collapses to FDAC* [Larrosa and Schiex, 2004]. When the constraints are binary and ternary, however, FDGAC* differs slightly from FDAC [Sanchez et al., 2008]. FDGAC* requires full supports with only zero unary costs, while FDAC [Sanchez et al., 2008] requires full supports with not only zero unary but also zero binary costs.

Based on the FDAC* Algorithm [Larrosa and Schiex, 2003], procedure $\text{enforceFDGAC*()}$ in Algorithm 5 enforces FDGAC*. Q and R store variables which are potentially not GAC* and not DGAC* respectively. Function $\text{popMax()}$ always removes the variable with the largest index from $R$ in constant time. Procedure $\text{enforceFDGAC*()}$ in Algorithm 5 must terminate, the proof of which is similar to those of Larrosa et al.’s Theorems 3 and 4 [2003]. Suppose $\text{findFullSupport()}$ and $\text{findSupport()}$ are of order $O(f_{DGAC})$ and $O(f_{GCAC})$ respectively, the complexity of procedure $\text{enforceFDGAC*()}$ can be stated as follows.
Theorem 7 enforceFDGAC*() has a time complexity of \( O(s^{2}_{\text{max}}(n f_{\text{FDGAC}} + f_{\text{GAC}}) + n^2d^2) \), where \( n, d, e, \) and \( s_{\text{max}} \) are defined in Theorem 1. Thus, enforceFDGAC*() must terminate.

Again, the complexity can be exponential due to findSupport() and findFullSupport(). In the following, we focus on findFullSupport(). The first part (lines 23 to 27) performs extension, a reversal of projection, to push all the unary costs back to \( C_S \). By the time we execute line 28, all unary costs are 0, and enforcing GAC* for \( x_i \) achieves the second requirement of DGAC*. Line 29 re-instantiates GAC* for all variables \( x_j \), where \( j > i \). Note that the success in line 28 guarantees that \( C_x(v_j) = 0 \) if \( v_j \) appears in a tuple \( \ell \) which makes \( C_S(\ell) = 0 \).

The key idea to performing extension properly is similar to that of projection: the method is applicable to a projection-safe soft global constraint \( C_S \) with cost function \( \mu \). Suppose now we want to extend a cost of \( \alpha \) associated with \( x_i = v \) from \( C_x \) to \( C_S \) resulting in a new cost function \( \mu'' \). In other words, \( \mu''(\ell) = \mu(\ell) \oplus \alpha \) if \( \ell[x_i] = v \); otherwise \( \mu''(\ell) = \mu(\ell) \). We construct the corresponding flow network of \( C_S \) with cost function \( \mu'' \) as \( G''(V, E, w'', c, s, t) \), where \( w'' = w_{\ell} \oplus \alpha \) if \( \ell \) is the edge corresponding to \( x_i = v \); otherwise \( w''_{\ell} = w_{\ell} \).

Similarly, extension is both sound and closed.

Theorem 8 Suppose \( C_S \) is a projection safe soft global constraint with cost function \( \mu \), and a cost of \( \alpha \) associated with \( x_i = v \) is extended from \( C_x \) to \( C_S \), resulting in a new cost function \( \mu'' \).

- (Soundness) If \( f \) is a minimum cost flow of \( G''(V, E, w'', c, s, t) \), then \( \sum_{\ell \in E} w''_{\ell} f_{\ell} = \min\{\mu''(\ell)\} \).
- (Closure) \( C_S \) with cost function \( \mu'' \) is projection safe.

The complexity result again follows from van Hoeve et al.’s Theorem 1 [2006].

Theorem 9 If \( C_S \) is a projection-safe soft global constraint, findFullSupport() has a time complexity of \( O(K + s_{\text{max}} \cdot SP) \), where \( K \) and \( SP \) are as defined in Theorem 2.

Last but not least, we state the relative strength of the consistencies concerned. Given two consistencies \( \beta \) and \( \gamma \), \( \beta \) is stronger than \( \gamma \) (\( \beta \geq \gamma \)) if a WCP \( P \) is \( \gamma \) whenever \( P \) is \( \beta \).

Theorem 10 FDGAC* \( \geq \) GAC* \( \geq \) strong \( \emptyset IC \geq \) GAC in “Soft as Hard” Approaches.

6 Experimental Results

To demonstrate the efficiency of our proposals, we have implemented strong \( \emptyset IC \), GAC*, and FDGAC* for the soft allDifferent constraint with the \( \mu_{\text{dec}} \) and \( \mu_{\text{var}} \) cost functions in ToulBar2.¹ Our benchmark instances are based on a softened version of the all-interval series problem (CSPLib Prob007). This problem contains mainly allDifferent constraints, unconcerning us from other possible external factors and focusing on evaluating the efficiency of our proposed algorithms. Such a benchmark also allows us to study the scaling behavior of our algorithms. The original problem of order \( n \) is to find a series \( \{x_1, \ldots, x_n\} \) such that it is a permutation of \( \{0, \ldots, n - 1\} \) and the adjacent differences \( d_i = |x_i - x_{i+1}| \), \( i = \{1, \ldots, n - 1\} \) are distinct. To model its softened version as a WCP, we use \( \{x_i\} \) and \( \{d_i\} \) as variables with domains \( \{0, \ldots, n - 1\} \). Two allDifferent constraints are placed on \( \{x_i\} \) and \( \{d_i\} \) respectively. Ternary table constraints are used to enforce \( d_i = |x_i - x_{i+1}| \). Besides, random unary constraints are placed on \( \{x_i\} \), assigning random costs to each assignment ranging from 0 to 9.

During the experiment, variables \( \{x_i\} \) are first assigned in lexicographic order, followed by \( \{d_i\} \) in the same order. Value assignments start with the value with minimum unary cost first. The test is conducted in a Dell Optiplex 280 with an Intel P4 3.2GHz CPU and 2GB RAM. The average runtime and number of backtracks of five instances are measured for each value of \( n \) with no initial upper bound. Entries are marked with a “*” if the average runtime exceeds the limit of 1 hour.

¹http://carlit.toulouse.inra.fr/cgi-bin/awki.cgi/ToolBarIntro
We give the results for allDifferent with $\mu_{dec}$ and $\mu_{var}$ in Figure 4, which agrees well with the theoretical comparison of the three consistencies. This demonstrates that minimum cost flow computation is an efficient method for enforcing the consistencies of projection-safe soft global constraints. Despite a higher complexity, FDGAC*, the strongest enforcing the consistencies of projection-safe soft global constraints, is the clear winner bettering strong consistency both in terms of pruning and lower bound reasoning, is the clear winner bettering strong consistency both in terms of pruning and lower bound reasoning, is the clear winner bettering strong consistency both in terms of pruning and lower bound reasoning, is the clear winner bettering strong consistency both in terms of pruning and lower bound reasoning, is the clear winner bettering strong consistency both in terms of pruning and lower bound reasoning, is the clear winner bettering strong consistency both in terms of pruning and lower bound reasoning, is the clear winner bettering strong consistency both in terms of pruning and lower bound reasoning, is the clear winner bettering strong consistency both in terms of pruning and lower bound reasoning.

Immediate future work includes studies of the implementation of more projection-safe soft global constraints, feasibility of other forms of consistencies, experiments on a wider variety of benchmarks. It is also interesting to investigate if there are other forms of projection-safety.

7 Conclusion

Global constraints are one of the keys for modeling and solving complex real-life problems. To the best of our knowledge, this is the first success report of global constraints in WCSP solvers with practical efficiency. Our techniques make it possible to enforce generalized versions of existing consistencies exploiting specifically characteristics of WSCFs.

We give the results for allDifferent with $\mu_{dec}$ and $\mu_{var}$ in Figure 4, which agrees well with the theoretical comparison of the three consistencies. This demonstrates that minimum cost flow computation is an efficient method for enforcing the consistencies of projection-safe soft global constraints. Despite a higher complexity, FDGAC*, the strongest consistency both in terms of pruning and lower bound reasoning, is the clear winner bettering strong $\Sigma IC$ by one to two orders of magnitude, while GAC* comes in a clear second. In the best case, enforcing FDGAC* can remove 18 times more search nodes than enforcing GAC*, and 220 times more than enforcing strong $\Sigma IC$. Last but not least, we note that, without strong $\Sigma IC$, Toulbar2 delays the propagation of $n$-ary constraints until only two variables restricted by the constraints are not yet assigned. It is impractical to solve the benchmark even with a small value of $n$.

References


