Polynomially Decomposable Global Cost Functions in Weighted Constraint Satisfaction

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

Abstract

In maintaining consistencies, such as GAC*, FDGAC* and weak EDGAC*, for global cost functions, Weighted CSP (WCSP) solvers rely on the projection and extension operations, which entail the computation of the cost functions' minima. Tractability of this minimum computation is essential for efficient execution. Since projections/extensions modify the cost functions, an important issue is *tractable projection-safety*, concerning whether minimum cost computation remains tractable after projections/extensions.

In this paper, we prove that tractable projection-safety is always *possible* for projections/extensions to/from the nullary cost function ($W_\varnothing$), and always *impossible* for projections/extensions to/from n-ary cost functions for $n \geq 2$. When $n = 1$, the answer is indefinite. We give a simple negative example, while Lee and Leung's flow-based projection-safe cost functions are also tractable projection-safe.

We propose *polynomially decomposable* cost functions, which are amenable to tractable minimum computation. We further prove that the polynomial decomposability property is unaffected by projections/extensions to/from unary cost functions. Thus, polynomially decomposable cost functions are tractable projection-safe. We show that the SOFT_AMONG, SOFT_REGULAR, SOFT_GRAMMAR and MAX_WEIGHT/MIN_WEIGHT are polynomially decomposable. They are embedded in a WCSP solver for extensive experiments to confirm the feasibility and efficiency of our proposal.

Introduction

Weighted Constraint Satisfaction Problems (WCSPs) give a framework for modeling and solving over-constrained and optimization problems. Besides being equipped with an efficient branch and bound procedure augmented with powerful consistency techniques, a practical WCSP solver should have a good library of global cost functions to model the often complex scenarios in real-life applications. Enforcing WCSP consistencies on a global cost function efficiently relies on two operations: (a) computing the minima of the cost functions and (b) projecting and/or extending costs among functions to create pruning opportunities. Global cost functions usually have high arities, but their special semantics enables specialized polynomial time algorithms for computing the minima. Unfortunately, projections/extensions modify a cost function so that its structure and even semantics might change, possibly making the original minimum computation algorithm no longer applicable. Therefore, the key notions here is *tractable projection-safety*, which concerns if the minimum computation of a projected/extended global cost function remains tractable.

In this paper, we first study tractable projection-safety in different scenarios of projections and extensions. We prove that a tractable cost function is always tractable projection-safe after projections/extensions to/from the nullary cost function ($W_\varnothing$), and always intractable after projections/extensions to/from n-ary cost functions for $n \geq 2$. When $n = 1$, the answer is indefinite. While flow-based projection-safe cost functions (Lee and Leung 2009; 2012) are positive examples of tractable projection-safe cost functions, we give a simple tractable global cost functions and show how it becomes intractable after projections/extensions to/from unary cost functions.

We introduce *polynomially decomposable* global cost functions, which can be decomposed into a polynomial number of simpler cost functions for (minimum) cost calculation. Computing minima of such cost functions, which is usually done by a polynomial time recursive memoization algorithm (or dynamic programming), is tractable and remains tractable after projections/extensions. Thus, polynomially decomposable cost functions are tractable projection-safe. Adding to the existing repertoire of global cost functions, we further prove that the global cost functions SOFT_AMONG, SOFT_REGULAR, SOFT_GRAMMAR, and MAX_WEIGHT/MIN_WEIGHT, are polynomially decomposable. To demonstrate the feasibility of our proposal, we implement and embed these cost functions in Toulbar2 and conduct experiments using four benchmarks to evaluate the performance with different consistency enforcements.

Background

A *WCSP* (Schiex, Fargier, and Verfaillie 1995) is a tuple $(\mathcal{X}, \mathcal{D}, \mathcal{C}, \top)$. $\mathcal{X}$ is a set of variables $\{x_1, x_2, \ldots, x_n\}$ ordered by their indices. Each variable has its finite domain $D(x_i) \in \mathcal{D}$ containing maximum d values that only one can assign to x_i. A tuple $\ell \in \mathcal{L}(S) = D(x_{s_1}) \times \ldots \times D(x_{s_n})$ is used to represent an assignment on $S = \{x_{s_1}, \ldots, x_{s_n}\} \subseteq$

$\mathcal{X}$. The notation $\ell[x_i]$ denotes the value assigned to x_i in ℓ, and $\ell[S']$ denotes the tuple formed from projecting ℓ onto $S' \subseteq S$. $\mathcal{C}$ is a set of cost functions. Each cost function $W_S \in \mathcal{C}$ has its scope $S \subseteq \mathcal{X}$, and maps $\ell \in \mathcal{L}(S)$ to a cost in the valuation structure $V(\top) = ([0 \dots \top], \oplus, \leq)$. $V(\top)$ contains a set of integers $[0 \dots \top]$ with standard integer ordering $\leq$. Addition $\oplus$ is defined by $a \oplus b = \min(\top, a + b)$. The subtraction $\ominus$ is defined only for $a \geq b$, as $a \ominus b = a - b$ if $a < \top$, and $\top$ otherwise. In particular, we define $W_\varnothing$ as the nullary cost function returning a constant cost, and W_i as the unary cost function over x_i. We assume the existence of $W_\varnothing$ and W_i for each variable x_i. Otherwise, we assume $W_i(v) = 0$ for $v \in D(x_i)$ and $W_\varnothing = 0$. The *cost* of a tuple $\ell \in \mathcal{L}(\mathcal{X})$ in a WCSP is defined as $cost(\ell) = W_\varnothing \oplus \bigoplus_{W_S \in \mathcal{C} \setminus \{W_\varnothing\}} W_S(\ell[S])$. Furthermore, a tuple ℓ is a *solution* of a WCSP if $cost(\ell)$ is minimum among all tuples in $\mathcal{L}(\mathcal{X})$.

WCSPs are usually solved by basic branch-and-bound search augmented with consistency techniques, which help pruning infeasible values and increase the value of $W_\varnothing$ while keeping the cost of all tuples in $\mathcal{L}(\mathcal{X})$ unchanged. Different consistency notions have been proposed such as NC* (Larrosa and Schiex 2004), AC* (Larrosa and Schiex 2004), FDAC* (Larrosa and Schiex 2003) and EDAC* (de Givry et al. 2005). Consistency enforcement usually involves three operations: (1) finding the minimum cost returned by the cost functions among all (or part of) tuples, (2) projecting costs to and (3) extending costs from smaller-arity cost functions. For simplicity, we write $min\{W_S(\ell) \mid \ell \in \mathcal{L}(S)\}$ as $min\{W_S\}$. We adopt definition of projection and extension from Cooper (2005) as follows. Given $S_2 \subset S_1$ and $|S_2| = r$. An r-*projection* of cost α from W_{S_1} to W_{S_2} with respect to $\ell \in \mathcal{L}(S_2)$ is a transformation of (W_{S_1}, W_{S_2}) to (W'_{S_1}, W'_{S_2}) such that:

$$W'_{S_1}(\ell') = \begin{cases} W_{S_1}(\ell') \ominus \alpha, & \ell'[S_2] = \ell \\ W_{S_1}(\ell'), & \text{otherwise} \end{cases}$$
$$W'_{S_2}(\ell') = \begin{cases} W_{S_2}(\ell') \oplus \alpha, & \ell' = \ell \\ W_{S_2}(\ell'), & \text{otherwise} \end{cases}$$

Extension is the reverse of projection. An r-*extension* of cost α from W_{S_2} to W_{S_1} with respect to $\ell \in L(S_2)$ is a transformation of (W_{S_1}, W_{S_2}) to (W'_{S_1}, W'_{S_2}) such that:

$$W'_{S_1}(\ell') = \begin{cases} W_{S_1}(\ell') \oplus \alpha, & \ell'[S_2] = \ell \\ W_{S_1}(\ell'), & \text{otherwise} \end{cases}$$
$$W'_{S_2}(\ell') = \begin{cases} W_{S_2}(\ell') \ominus \alpha, & \ell' = \ell \\ W_{S_2}(\ell'), & \text{otherwise} \end{cases}$$

Note that we allow $S_2 = \varnothing$ in the definition, which is a projection to or an extension from $W_\varnothing$.

A *global cost function* is a cost function with special semantics, based on which efficient algorithms can be designed for consistency enforcements. In particular, we denote a global cost function as GC^μ_S if it is derived from the corresponding hard global constraint GC with a violation measure μ and variable scope S. GC^μ_S returns how much a tuple on S has violated the original hard constraint, or 0 if the tuple satisfies the constraint.

Tractable Projection Safety

A global cost function W_S is *tractable* if computing $\min\{W_S\}$ is tractable under standard integer operations. Furthermore, W_S is *tractable r-projection-safe* if (a) W_S is tractable, and (b) $\Delta_r(W_S)$ is tractable where Δ_r is a finite sequence of r-projection and/or r-extension from/to other cost functions W'_{S}, where $|S'| = r \leq |S|$, and $\Delta_r(W_S)$ is the cost function after applying Δ_r. Note that, in practice, we do not enforce the costs not exceeding $\top$ since it is expensive and tedious to maintain. When it comes to pruning, we treat any cost beyond $\top$ practically as $\top$.

We divide the discussion of tractable r-projection-safety into three cases: (a) $r = 0$, (b) $r \geq 2$ and (c) $r = 1$.

When $r = 0$, projections and extensions are only to/from $W_\varnothing$, which are employed in $\varnothing$IC (Zytnicki, Gaspin, and Schiex 2009) and strong $\varnothing$IC (Lee and Leung 2009; 2012) enforcement. If a cost function is tractable, it remains tractable after applying Δ_0, since $\Delta_0(W_S)$ and W_S differ only by a constant.

Observation 1 *If a cost function W_S is tractable, it is tractable 0-projection-safe.*

When $r \geq 2$, projections and extensions are to/from r-arity cost functions, and required for enforcing consistencies in ternary cost functions (Sanchez, de Givry, and Schiex 2008) and complete k-consistency (Cooper 2005). In general, even if a cost function W_S is tractable, W_S may not be tractable r-projection-safe for $r \geq 2$.

Theorem 1 *If a global cost function W is tractable, it is not tractable r-projection-safe for $r \geq 2$.*

Proof: We show that if W is tractable r-projection-safe for $r \geq 2$, then solving a general CSP $(\mathcal{X}, \mathcal{D}, \mathcal{C}^h)$ becomes tractable, which is a contradiction. Without loss of generality, we assume the scope S of all constraints in $C^h_S \in \mathcal{C}^h$ has size r. We define a WCSP P and construct a particular Δ_r based on P such that the CSP can be solved by $\Delta_r(W_\mathcal{X})$. P is defined as $(\mathcal{X}, \mathcal{D}, \mathcal{C} \cup \{W_\mathcal{X}\}, \top)$ based on the given CSP as follows. We define $\top$ as a sufficiently large integer such that $\top > W_\mathcal{X}(\ell)$ for every $\ell \in \mathcal{L}(\mathcal{X})$. $\mathcal{C}$ contains $|\mathcal{C}^h|$ cost functions. $W_S \in \mathcal{C}$ is defined from C^h_S as $W_S(\ell) = 0$ if ℓ is accepted by C^h_S, or $\top$ otherwise. From the WCSP, Δ_r can be defined as follows: for each forbidden tuple $\ell[S]$ in each $C^h_S \in \mathcal{C}^h$, we add an extension of $\top$ from W_S to $W_\mathcal{X}$ with respect to $\ell[S]$ to Δ_r. Under this construction, $\Delta_r(W_\mathcal{X})(\ell)$ is equivalent to $W_\mathcal{X}(\ell) \oplus \bigoplus_{W_S \in \mathcal{C}} W_S(\ell[S])$. If a tuple $\ell \in \mathcal{L}(\mathcal{X})$ contains a tuple $\ell[S]$ forbidden by $C^h_S \in \mathcal{C}^h$, $\Delta_r(W_\mathcal{X})(\ell) \geq \top$ because of extension. Thus, $\min\{\Delta_r(W_\mathcal{X})\} \geq \top$ implies that the given CSP is unsatisfiable. Because solving CSPs is NP-Hard, $\Delta_r(W_\mathcal{X})$ cannot be tractable. ∎

When $r = 1$, we have 1-projections and 1-extensions which are the backbone of the consistency algorithms of (G)AC* (Larrosa and Schiex 2004; Lee and Leung 2009; 2012), FD(G)AC* (Larrosa and Schiex 2003; Lee and Leung 2009; 2012) and (weak) ED(G)AC* (de Givry et al. 2005; Lee and Leung 2010; 2012). Tractable cost functions are tractable 1-projection-safe only under special conditions. For example, Lee and Leung (2009; 2012) propose sufficient conditions for tractable cost function to be (flow-based) projection-safe.

A global cost function W_S is flow-based if it can be represented by a flow network G such that the minimum cost flow on G corresponds to $\min\{W_S\}$ (van Hoeve, Pesant, and Rousseau 2006). Lee and Leung (2009; 2012) prove that a flow-based projection-safe cost function is flow-based, and it is still flow-based after 1-projections and 1-extensions. Flow-based projection-safety implies tractable 1-projection-safety. We state as the following theorem.

Theorem 2 *(Lee and Leung 2009; 2012) If a cost function is flow-based projection-safe, it is tractable* 1*-projection-safe.*

We also observed that tractable cost functions are not necessarily tractable 1-projection-safe. One example is $2SAT^{\text{const}}$. Given a set of boolean variables S, a set of binary clauses F, and a positive integer c. The cost function $2SAT^{\text{const}}(S, F, c)$ returns 0 if the assignments on S satisfies F, or c otherwise. $2SAT^{\text{const}}$ is tractable, because `2SAT` problem is tractable (Krom 1967). However, it is not tractable 1-projection-safe.

Theorem 3 $2SAT^{\text{const}}$ *is not tractable* 1*-projection-safe.*

Proof: We show that if $2SAT^{\text{const}}(\mathcal{X}, F, k)$ is tractable 1-projection-safe, the NP-Hard problem $\texttt{W2SAT}(\mathcal{X}, F, k)$ is tractable. Given a set of boolean variable $\mathcal{X}$, a set of binary clauses F, and a fixed integer k. $\texttt{W2SAT}(\mathcal{X}, F, k)$ determines whether there exists an assignment on $\mathcal{X}$ such that at most k variables in $\mathcal{X}$ is set to $true$ and satisfies all clauses in F (Creignou, Khanna, and Sudan 2001).

We construct a particular Δ_1 such that $\texttt{W2SAT}(\mathcal{X}, F, k)$ can be solved from $W_{\mathcal{X}} = 2SAT^{\text{const}}(\mathcal{X}, F, k)$ from the boolean WCSP $P = (\mathcal{X}, \mathcal{D}, \mathcal{C} \cup \{W_{\mathcal{X}}\}, k + 1)$. $\mathcal{C}$ only contains unary cost functions W_i for each $x_i \in \mathcal{X}$, which returns 1 if $x_i = true$ and 0 otherwise. Based on P, we construct Δ_1 as follows: for each variables $x_i \in \mathcal{X}$, we add an extension of 1 from W_i to $W_{\mathcal{X}}$ with respect to the value $true$ into Δ_1. As a result, the tuple ℓ with $\Delta_1(W_{\mathcal{X}})(\ell) = k' \le k$ ensures that k' variables in ℓ is set to $true$ and satisfying F, because $x_i = true$ will incur a cost of 1. Thus, the `W2SAT` has a solution iff $\min\{\Delta_1(W_{\mathcal{X}})\} \le k$. However, `W2SAT` is not tractable (Creignou, Khanna, and Sudan 2001), so is $\Delta_1(W_{\mathcal{X}})$. ■

When the context is clear, we refer tractable 1-projection-safety, 1-projection and 1-extension to simply as tractable projection-safety, projection and extension respectively hereafter.

Polynomial Decomposability

Lee and Leung (2009; 2012) give one class of tractable projection-safe cost functions. In this section, we introduce an additional class, as inspired by dynamic programming algorithms and based on a decomposition of global cost functions.

A cost function W_S can be *safely decomposed* to a set of cost functions $\Omega = \{\omega_{S_1}, \ldots, \omega_{S_m}\}$ using cost aggregation function f, where $S_i \subseteq S$, iff

1. $W_S(\ell) = f(\{\omega_{S_i}(\ell[S_i]) \mid \omega_{S_i} \in \Omega\})$
2. f is *distributive, i.e.*
   (a) $\min\{W_{S_i}\} = f(\{\min\{\omega_{S_i}\} \mid \omega_{S_i} \in \Omega\})$, and;
   (b) For a variable $x \in S$, a cost α and a tuple $\ell \in \mathcal{L}(S)$, $W_S(\ell) \oplus \alpha = f(\{\omega_{S_i}(\ell[S_i]) \oplus \nu_{x,S_i}(\alpha) \mid \omega_{S_i} \in \Omega\})$ and $W_S(\ell) \ominus \alpha = f(\{\omega_{S_i}(\ell[S_i]) \ominus \nu_{x,S_i}(\alpha) \mid \omega_{S_i} \in \Omega\})$, where the function ν is defined as $\nu_{x,S_i}(\alpha) = \alpha$ if $x \in S_i$, and 0 otherwise.

In other words, W_S can be represented as a combination of Ω. A distributive f implies that (a) $\min(W_S)$ can be computed from $\min\{\omega_{S_i}\}$ for $i \in \{1, ..., n\}$ and (b) projections/extensions on W_S can be distributed to its components. We state the latter directly as the following theorem: define $\delta_{x_i,v}(W_S)$ as the cost function from W_S after applying a projection to or extension from W_i with respect to $v \in D(x_i)$ if $x_i \in S$, or W_S if $x_i \notin S$.

Theorem 4 *If W_S can be safely decomposed to Ω using f, then $\delta_{x_i,v}(W_S)$ can also be safely decomposed to $\Omega' = \{\delta_{x_i,v}(\omega_{S_1}), \ldots, \delta_{x_i,v}(\omega_{S_m})\}$ using f.*

Safely decomposable cost functions may not be tractable. We further give conditions for tractability. A (global) cost function W_S can be *polynomially decomposed* into a set of cost functions $\Omega = \{\omega_{S_1}, \ldots, \omega_{S_m}\}$, where $S_i \subseteq S$, if

1. m is polynomial in the size of S and and maximum domain size d,
2. Each $\omega_{S_i} \in \Omega \cup \{\omega_{S_{m+1}}\}$, where $\omega_{S_{m+1}} = W_S$, is either a tractable unary cost function, or can be safely decomposed into $\Omega_i \subseteq \{\omega_{S_j} \mid j < i\}$ using a tractable cost aggregation function f_i.

Polynomially decomposable cost functions are tractable and also tractable projection-safe as stated below.

Theorem 5 *A polynomially decomposable cost function W_S is tractable.*

Proof: Algorithm 1 can be applied to compute $min\{W_S\}$. The algorithm uses a recursion approach with memoization. A table `Min` is used to store minimum costs of each cost function to avoid re-computation. Thus, $\min\{\omega_{S_i}\}$ is evaluated only once in polynomial time. The time complexity of Algorithm 1 is based on the worst-case time complexity of computing each tractable function f_i. Result follows. ■

```
Function ComputeMin(W_S)
    foreach 1 ≤ i ≤ m + 1 do Min [ω_{S_i}] := NULL ;
    return Eval (ω_{S_{m+1}});

Function Eval(ω_{S_i})
    if Min [ω_{S_i}] = NULL then
        if Ω_i = ∅ or |S_i| ≤ 1 then
            Min [ω_{S_i}] := min{ω_{S_i}};
        else
            P_i := ∅;
            foreach ω_{S_j} ∈ Ω_i do
                P_i := P_i ∪ {Eval(ω_{S_j})};
            Min [ω_{S_i}] := f_i(P_i);
    return Min [ω_{S_i}];
```

Algorithm 1: Compute $\min\{W_S\}$

The following lemma is useful in proving our final result.

Lemma 1 *Suppose W'_S is the cost function from W_S after applying a projection to or a extension from W_i, where $x_u \in S$, with respect to $v \in D(x_i)$. If W_S is polynomially decomposable, so is W_S.*

Proof: We only prove on the part of projection, while the proof on extension is similar.

Consider a cost function $\omega_{S_i} \in \Omega \cup \{\omega_{S_{m+1}}\}$, where $\omega_{S_{m+1}} = W_S$. If ω_{S_i} is a tractable unary cost function, then after applying projection on ω_{S_i}, the resultant cost function is $\omega'_{S_i}(\ell) = \omega_{S_i}(\ell) \ominus \nu_{x_u,S_i}(\alpha)$, which is still tractable; if ω_{S_i} can be safety decomposed into $\Omega_i \subseteq \{\omega_{S_1}, \ldots, \omega_{S_{i-1}}\}$ using f_i, by Theorem 4, the resultant function ω'_S after projection can also be safety decomposed into $\Omega'_i \subseteq \{\omega'_{S_1}, \ldots, \omega'_{S_{i-1}}\}$ using f_i. Since m and each f_i are unchanged, W'_S can be polynomially decomposed into a $\{\omega'_{S_1}, \ldots, \omega'_{S_m}\}$. Result follows. ■

Directly from Lemma 1, W_S is tractable projection-safe, as stated below.

Theorem 6 *If W_S is polynomially decomposable, it is tractable projection-safe.*

We present another useful class of projection-safe global functions. Algorithm 1 also gives an efficient algorithm to compute the minimum cost. In the following, we give examples of this class of cost functions.

Examples

Checking if a cost function W_S is polynomially decomposable amounts to finding a polynomially sized set of simpler cost functions that can be combined using a distributive cost aggregation function to compute W_S. In the following, we state without proof a distributive aggregation function for use in the rest of this section.

Lemma 2 *If a global cost function W_S can be represented as $W_S(\ell) = \min_{i=1}^{r}\{\bigoplus_{j=1}^{n_i} \omega_{S_{i,j}}(\ell[S_{i,j}])\}$, where:*

- *$\sum_{i=1}^{r} n_i$ is polynomial in $|S|$ and d, and;*
- *for each i, $S_{i,j} \cap S_{i,k} = \varnothing$ iff $j \neq k$ and $\bigcup_{j}^{n_i} S_{i,j} = S$,*

then W_S is safely decomposable.

In the following, we give five examples of polynomially decomposable cost functions. They are SOFT_AMONG, SOFT_REGULAR, SOFT_GRAMMAR, MAX_WEIGHT and MIN_WEIGHT cost functions. For simplicity, we assume the scope of each global cost function is $S = \{x_1, \ldots, x_n\}$.

The SOFT_AMONG Cost Function

Given a set of values V, the lower bound lb and the upper bound ub such that $0 \leq lb \leq ub \leq |S|$. Define $t(\ell) = |\{i \mid \ell[x_i] \in V\}|$. The SOFT_AMONG$^{var}(S, lb, ub, V)$ returns $max(0, lb - t(\ell), t(\ell) - ub)$ (Solnon et al. 2008).

Theorem 7 *The SOFT_AMONG^{var} cost function is polynomially decomposable and thus tractable projection-safe.*

Proof:

Assume W_S = SOFT_AMONG$^{var}(S, lb, ub, V)$. Define $\omega^j_{S_i}$ = SOFT_AMONG$^{var}(S_i, j, j, V)$, where $S_i = \{x_1, \ldots, x_i\} \subseteq S$. Each $\omega^j_{S_i}$ can be represented recursively as follows.

Define $f_\omega(i, j, \ell) = \omega^j_{S_i}(\ell)$. Each $f_\omega(i, j, \ell)$ can be represented as follows:

$$\begin{array}{lcl} f_\omega(0, j, \ell) & = & j \\ f_\omega(i, 0, \ell) & = & f_\omega(i-1, 0, \ell[S_{i-1}]) \oplus \overline{U}_i^V(\ell[x_i]) \\ & & \text{for } i > 0 \\ f_\omega(i, j, \ell) & = & \min \left\{ \begin{array}{l} f_\omega(i-1, j-1, \ell[S_{i-1}]) \oplus U_i^V(\ell[x_i]) \\ f_\omega(i-1, j, \ell[S_{i-1}]) \oplus \overline{U}_i^V(\ell[x_i]) \end{array} \right. \\ & & \text{for } j > 0, i > 0 \end{array}$$

The function $U_i^V(v)$, where $v \in D(x_i)$, returns 0 if $v \in V$, or 1 otherwise, while the function $\overline{U}_i^V(v)$ returns 0 if $v \notin V$, or 1 otherwise. For all tuples $\ell \in \mathcal{L}(S)$, $W_S(\ell) = \min_{lb \leq j \leq ub}\{f_\omega(n, j, \ell)\}$. By Lemma 2, SOFT_AMONG$^{var}_S$ is safely decomposable, so is $\omega(S_i, j)$. By Theorem 6, results follow. ■

From the formulation in Theorem 7, the time complexity of enforcing GAC* on SOFT_AMONG^{var} can be stated as follows:

Theorem 8 *Enforcing GAC* on SOFT_AMONG^{var} requires $O(nd(n^2 + nd))$, where $n = |S|$ and $d = \max_{x_i \in S}\{|D(x_i)|\}$.*

The SOFT_REGULAR Cost Function

A regular language $L(M)$ can be represented by a finite state automaton (DFA) $M = (Q, \Sigma, \delta, q_0, F)$. Q is a set of states. Σ is a set of characters. The transition function δ is defined as: $\delta : Q \times \Sigma \mapsto Q$. $q_0 \in Q$ is the initial state, and $F \subseteq Q$ is the set of final states. A string $\tau \in L(M)$ if τ can lead the transitions from q_0 to $q_f \in F$ in M. The constraint REGULAR$_S(M)$ accepts a tuple $\ell \in \mathcal{L}(S)$ if $\tau_\ell \in L(M)$, where τ_ℓ is the string formed from ℓ (Pesant 2004). The corresponding soft variant SOFT_REGULAR$^{var}(S, M)$ returns $\min\{H(\tau_\ell, \tau_i) \mid \tau_i \in L(M)\}$, where $H(\tau_1, \tau_2)$ returns the Hamming distance between τ_1 and τ_2 (Beldiceanu, Carlsson, and Petit 2004; van Hoeve, Pesant, and Rousseau 2006).

Theorem 9 *SOFT_REGULAR^{var} is polynomially decomposable and thus tractable projection-safe.*

Proof: Results follow directly from the directed DAG representation of SOFT_REGULAR^{var} (Beldiceanu, Carlsson, and Petit 2004; van Hoeve, Pesant, and Rousseau 2006), Lemma 2 and Theorem 6. ■

Note that the result collides with Theorem 18 by Lee and Leung (2012). The time complexity of enforcing GAC* on SOFT_REGULAR^{var} can be stated as follows:

Theorem 10 *Enforcing GAC* on SOFT_REGULAR^{var} requires $O(nd(nd \cdot |Q|))$, where n and d are defined in Theorem 8.*

Note that the time complexity involves $|Q|$, which can possibly dominate over n and d in some cases. In practice, $|Q|$ is typically a constant, or polynomial in n and d, which does not dominate the run-time.

Theorem 10 also gives another proof of the tractable 1-projection-safety of SOFT_AMONG cost functions. The size of the DFA representing the AMONG constraint is polynomial in n.

The SOFT_GRAMMAR Cost Function

A context-free language $L(G)$ can be represented as a context-free grammar $G = (\Sigma, N, P, A_0)$. Σ is a set of characters called terminals. N is a set of symbols called non-terminals. P is a set of production rules from N to $(\Sigma \cup N)^*$, where $*$ is the Kleene star. $A_0 \in N$ is a starting symbol. A string $\tau \in L(G)$ iff τ can be derived from G. The constraint $\text{GRAMMAR}(S, G)$ accepts a tuple $\ell \in \mathcal{L}(S)$ if $\tau_\ell \in L(G)$ (Kadioglu and Sellmann 2010). Its soft variant $\text{SOFT\_GRAMMAR}^{var}(S, G)$ returns $\min\{H(\tau_\ell, \tau_i) \mid \tau_i \in L(G)\}$ respectively (Katsirelos, Narodytska, and Walsh 2011).

Theorem 11 $\text{SOFT\_GRAMMAR}^{var}(S, G)$ *is polynomially decomposable and thus tractable projection-safe.*

Proof: Results follow from a direct adaptation of the modified CYK parsing algorithm (Katsirelos, Narodytska, and Walsh 2011) which is based on dynamic programming, Lemma 2 and Theorem 6. ■

The time complexity can be stated as follows.

Theorem 12 *Enforcing GAC* on* $\text{SOFT\_GRAMMAR}^{var}_S$ *requires* $O(nd((n^3 + nd) \cdot |P|))$. n *and* d *are defined in Theorem 8.*

Again, in practice, $|P|$ is a constant, or polynomial in n and d,

Again, the time complexity involves $|P|$, which does not dominate the run-time.

Note that Theorem 12 gives another proof of showing that SOFT_AMONG and SOFT_REGULAR are tractable 1-projection-safe if the number of states in the DFA is constant, or polynomial in n and d. A DFA with polynomial number of states can be transformed into a grammar with polynomial number of production rules.

The MAX_WEIGHT/MIN_WEIGHT Cost Functions

Given a weight function $c(x_i, v)$ that maps a variable-value pair to a fixed cost. The $\text{MAX\_WEIGHT}(S, c)$ function returns $\max\{c(x_i, \ell[x_i]) \mid x_i \in S\}$, while the $\text{MIN\_WEIGHT}(S, c)$ function returns $\min\{c(x_i, \ell[x_i]) \mid x_i \in S\}$ for each tuple $\ell \in \mathcal{L}(S)$. Note that they can be regarded the weighted version of the MAXIMUM/MINIMUM hard constraints (Beldiceanu 2001).

Theorem 13 $\text{MAX\_WEIGHT}(S, c)$ *and* $\text{MIN\_WEIGHT}(S, c)$ *are polynomially decomposable, and thus tractable projection-safe.*

Proof: Note that direct decomposition from definition is unsafe. In the following, we give a safe decomposition of $\text{MAX\_WEIGHT}(S, c)$, while that of $\text{MIN\_WEIGHT}(S, c)$ is similar.

We define two unary functions H_i^u and $G_i^{j,u}$. $H_i^u(v)$ returns $c(x_i, v)$ if $v = u$ and $\top$ otherwise. $G_i^{j,u}$ returns 0 if $c(x_i, v) \leq c(x_j, u)$ and $\top$ otherwise. They give a safe decomposition for MAX_WEIGHT as follows:

$$\text{MAX\_WEIGHT}_S(c)(\ell) = \min_{x_i \in S, v \in D(x_i)} \{H_i^v(\ell[x_i]) \oplus \bigoplus_{x_j \in S \setminus \{x_i\}} G_j^{i,v}(\ell[x_j])\}$$

H_i^v represents the choice of the maximum weighted component in the tuple, while $G_j^{i,v}$ represents the choice of each component other than the one with the maximum weight. By Lemma 2 and Theorem 6, results follow. ■

Theorem 14 *Enforcing GAC* on* $\text{MAX\_WEIGHT}(S, c)$ *or* $\text{MIN\_WEIGHT}(S, c)$ *requires* $O(nd(nd \cdot \log(nd)))$, *where* n *and* d *are defined in Theorem 8.*

Note that the time complexity results stated above assume that we compute the minimum every time we need to find the support for each variable. In practice, the computation can be incremental, thus with a lower time complexity.

In the next section, we put theory into practice. We demonstrate our framework with different benchmarks and compare results with different consistency notions.

Experiments

In this section, we put theory into practice, by implementing the cost functions described in the previous section in Toulbar2 v0.9 to demonstrate the practicality of our algorithmic framework in solving over-constrained and optimization problems. We also compare the results with strong ∅IC, GAC* and FDGAC*, which have covered all three cases: 0-projections, 1-projections and 1-extensions.

In the experiments, variables are assigned in lexicographic order. Value assignment starts with the value with the minimum unary cost. The tests are conducted on an Intel Core2 Duo E7400 (2 x 2.80GHz) machine with 4GB RAM. Each benchmark has a different timeout. We first compare the number of solved instances. Among the solved instances, we report and compare their average run-time and number of backtracks. Out of 10 randomly generated test cases of each parameter setting, the best results are marked using the '†' symbol.

The Car Sequencing Problem The problem (CSPLib prob001) (Parrello, Kabat, and Wos 1986) consists of sequencing n cars specified with different options on an assebly line. The model consists of n variables and counting requirements. We generate over-constrained instances and soften the model using the SOFT_AMONG^{var} and SOFT_GCC^{var} cost functions.

Table 1 gives the experimental results. The results show that enforcing FDGAC* reduces the number of backtracks at least 25 times more than strong ∅IC, and 10 times more than GAC*. Besides, enforcing FDGAC* runs at least 1.2 times faster than GAC*, and 4 times faster than strong ∅IC.

Table 1: Car sequencing problem (timeout=5min)

n	strong ∅IC			GAC*			FDGAC*		
	solved	time	backtracks	solved	time	backtracks	solved	time	backtracks
14	†10	42.84	234537	†10	16.80	67842	†10	†1.37	†1607
15	8	136	715754	†10	29.75	109085	†10	†4.49	†4978
16	3	178.98	834998	8	133.08	434969	†10	†6.90	†6179
17	1	163.73	830343	2	130.14	387446	†10	†48.07	†35218

The Nonogram Problem The problem (CSPLib prob012) (Ishida 1994) is a typical board puzzle to shade blocks in an

$n \times n$ board. The requirements involve patterns. The model consists of n^2 variables and SOFT_REGULAR^{var} cost functions.

Table 2 shows the results of the experiment. In a time limit of 5 minutes, enforcing strong ∅IC could only solve relatively small instances ($n = 6$). Enforcing GAC* could solve relatively larger ones ($n = 8$). For $n = 10$, all instances can only be solved when FDGAC* is enforced, and each instance is solved within a minute.

Table 2 also gives the comparison on the SOFT_REGULAR^{var} function when it is enforced by polynomially decomposable approach and flow-based projection-safe approach (Lee and Leung 2009; 2012). The two approaches result in the same search tree when we enforce the same consistency, but the run-time varies. Results show that using the polynomially decomposable approach speeds up searching by at least 3 times than using the flow-based projection-safe approach, due to the large constant factor behind the flow algorithm.

Table 2: Nonogram (timeout=5min)

polynomially decomposable approach									
n	strong ∅IC			GAC*			FDGAC*		
	solved	time	backtrack	solved	time	backtrack	solved	time	backtrack
6	†10	9.50	150167	†10	0.03	763	†10	†0.00	†109
7	1	245.17	2627322	†10	3.88	72811	†10	† 0.03	†345
8	0	*	*	7	113.76	1730882	†10	†0.12	†842
9	0	*	*	2	52.85	764467	†10	†0.34	†1500
10	0	*	*	0	*	*	†10	†11.78	†22828
flow-based approach									
n	strong ∅IC			GAC*			FDGAC*		
	solved	time	backtrack	solved	time	backtrack	solved	time	backtrack
6	9	25.23	72130	†10	0.32	763	†10	0.05	109
7	0	*	*	†10	60.84	72811	†10	0.48	345
8	0	*	*	1	26.59	28166	†10	2.04	842
9	0	*	*	1	151.38	83479	†10	5.87	1500
10	0	*	*	0	*	*	9	40.67	4848

Well-formed Parentheses Given a set of $2n$ even length intervals in $[1, \ldots, 2n]$. The problem is to find a string of parentheses with length $2n$ such that substrings in each of the intervals are well-formed parentheses. We model this problem by a set of $2n$ variables. We post a SOFT_GRAMMAR^{var} cost function for each interval to represent the requirement of well-formed parentheses.

Results are shown in Table 3. Enforcing FDGAC* reduces the search space 6 times and thus speeds up the search 20 times more than ∅IC. Enforcing FDGAC* always outperforms GAC* by up to 6 times in terms of search space reduction and 3 times in term of runtime.

Table 3: Well-formed parentheses (timeout=5min)

n	strong ∅IC			GAC*			FDGAC*		
	solved	time	backtracks	solved	time	backtracks	solved	time	backtracks
10	†10	6.36	5552	†10	0.54	408	†10	†0.43	†172
11	†10	17.38	10253	†10	1.48	784	†10	†0.92	†245
12	†10	47.19	22668	†10	3.38	1383	†10	†2.08	†394
13	9	90.94	34435	†10	6.98	2175	†10	†2.89	†440
14	4	176.1	59756	†10	31.99	7208	†10	†7.75	†765
15	0	*	*	†10	56.43	9705	†10	†15.59	†1026
16	0	*	*	†10	85.58	14825	†10	†20.12	†1367
17	0	*	*	6	158.16	25546	†10	†54.94	†3346

The Minimum Energy Broadcasting Problem The task (CSPLib prob048) (Burke and Brown 2007) is to find a broadcast tree that minimizes the sum of the energy consumed by n wireless routers in a network. The model consists of n variables, and one hard global constraint TREE (Beldiceanu, Carlssoon, and Rampon 2005) and MAX_WEIGHT$(\mathcal{X}, c_i)$ cost functions. We also implement the GAC enforcement algorithm of the TREE constraint from Beldiceanu *et al.* (2005).

Results are shown in Table 4, which is different from the previous experiments. FDGAC* can reduce the search spaces up to 6 times more than GAC*, but runs 2 times slower than GAC*. The hard TREE global constraint accounts for the results, which can only achieve strong ∅IC and GAC* but not FDGAC*.

Table 4: Minimum energy broadcast (timeout=10min)

n	m	strong ∅IC			GAC*			FDGAC*		
		solved	time	backtrack	solved	time	backtrack	solved	time	backtrack
20	40	†10	8.03	61806	†10	†1.64	9080	†10	2.03	†1352
20	60	†10	26.08	153237	†10	†13.54	55317	†10	37.77	†16694
20	100	†10	13.55	69453	†10	†12.50	37323	†10	41.78	†12106
25	50	†10	72.55	303422	†10	†15.34	52855	†10	15.48	†4849
25	75	5	301.68	1044058	†7	†229.10	625415	5	176.45	†34108
25	125	†5	50.27	121473	†5	†43.04	73262	3	166.85	†22005
30	60	4	216.44	557575	†9	†101.33	233610	†9	118.48	†21424
30	90	1	401.92	1050414	†2	†162.63	293660	1	305.96	†43238

Conclusion

Bessiere *et al.* (2011) suggest another decomposition of global cost functions into simpler functions in table form based on the Berge-acyclic property. Their example is based on the REGULAR global constraint, and the decomposition is one level directly onto ternary functions. In the case of a polynomially decomposable cost function, the decomposition can be recursive, which is amenable to efficient minimum cost computation utilizing dynamic programming techniques.

Our contributions are four-fold. First, we define the *tractable r-projection-safety* property, and study the property with respect to projections/extensions with different arities of cost functions. We show that projection-safety is always possible for projections/extension to/from the nullary cost function, while it is alway impossible for projections/extensions to/from r-ary cost functions for $r \geq 2$. When $r = 1$, we show that a tractable cost function may or may not be tractable 1-projection-safe. Second, we define *polynomially decomposable cost functions* and show them to be tractable 1-projection-safe. We give also a polytime dynamic programming based algorithm to compute the minimum cost of this class of cost functions. Third, we further show that the cost functions SOFT_AMONG^{var}, SOFT_REGULAR^{var}, SOFT_GRAMMAR^{var}, and MAX_WEIGHT/MIN_WEIGHT, are polynomially decomposable and tractable 1-projection-safe. Fourth, we perform experiments and compare typical WCSP consistency notions and shows that our algorithm framework works well with GAC* and FDGAC* enforcement, in terms of run-time and reduction in search space. We also compare against the flow-based approach (Lee and Leung 2009; 2012) and show that our approach is more competitive.

The concept of polynomial decomposability is inspired by a simple dynamic programming approach. It is unclear if we can go beyond dynamic programming and yet maintain tractability. An immediate possible future work is to characterize exactly the class of polynomial decomposable cost functions. It will also be interesting to investigate other forms of sufficient conditions for polynomial decomposability. The sufficient conditions given in Lemma 2 provides only a partial answer.

On the practical side, besides polynomial decomposability and flow-based project-safety (Lee and Leung 2009; 2012), we would like to investigate other forms of tractable 1-projection-safety and techniques for enforcing typical consistency notions efficiently. It is also interesting to investigate possibility to apply our framework to other stronger consistency notions such as VAC (Cooper et al. 2010).

Acknowledgements

We are grateful to comments and suggestions by Thomas Schiex and the anonymous referees. The work described in this paper was generously supported by grants CUHK413808 and CUHK413710 from the Research Grants Council of Hong Kong SAR.

References

Beldiceanu, N.; Carlsson, M.; and Petit, T. 2004. Deriving Filtering Algorithms from Constraint Checkers. In *Proceedings of CP'04*, 107–122.

Beldiceanu, N.; Carlssoon, M.; and Rampon, J. 2005. The Tree Constraint. In *Proceedings of CPAIOR'05*, 64–75.

Beldiceanu, N. 2001. Pruning for the Minimum Constraint Family and for the Number of Distinct Values Constraint Family. In *Proceedings of CP'01*, 211–224.

Bessiere, C.; Boizumault, P.; de Givry, S.; Gutierrez, P.; Loudni, S.; Metivier, J.; and Schiex, T. 2011. Decomposing Global Cost Functions. In *Proceedings of Soft'11 Workshop (at CP'11)*.

Burke, D., and Brown, K. 2007. Using Relaxations to Improve Search in Distributed Constraint Optimization. *Artificial Intelligence Review* 28(1):35–50.

Cooper, M.; de Givry, S.; Sanchez, M.; Schiex, T.; Zytnicki, M.; and Werner, T. 2010. Soft Arc Consistency Revisited. *Artificial Intelligence* 174:449–478.

Cooper, M. C. 2005. High-Order Consistency in Valued Constraint Satisfaction. *Constraints* 10(3):283–305.

Creignou, N.; Khanna, S.; and Sudan, M. 2001. Complexity Classifications of Boolean Constraint Satisfaction Problems. *SIAM Monographs on Discrete Mathematics and Applications* 7.

de Givry, S.; Heras, F.; Zytnicki, M.; and Larrosa, J. 2005. Existential Arc Consistency: Getting Closer to Full Arc Consistency in Weighted CSPs. In *Proceedings of IJCAI'05*, 84–89.

Ishida, N. 1994. Game "NONOGRAM"(in Japanese). *Mathematical Seminar* 10:21–22.

Kadioglu, S., and Sellmann, M. 2010. Grammar Constraints. *Constraints* 15(1):117–144.

Katsirelos, G.; Narodytska, N.; and Walsh, T. 2011. The Weighted GRAMMAR Constraints. *Annals of Operations Research* 184(1):179–207.

Krom, M. 1967. The Decision Problem for a Class of First-Order Formulas in Which all Disjunctions are Binary. *Mathematical Logic Quarterly* 13(1-2):15–20.

Larrosa, J., and Schiex, T. 2003. In the Quest of the Best Form of Local Consistency for Weighted CSP. In *Proceedings of IJCAI'03*, 239–244.

Larrosa, J., and Schiex, T. 2004. Solving Weighted CSP by Maintaining Arc Consistency. *Artificial Intelligence* 159(1-2):1–26.

Lee, J. H. M., and Leung, K. L. 2009. Towards Efficient Consistency Enforcement for Global Constraints in Weighted Constraint Satisfaction. In *Proceedings of IJCAI'09*, 559–565.

Lee, J. H. M., and Leung, K. L. 2010. A Stronger Consistency for Soft Global Constraints in Weighted Constraint Satisfaction. In *Proceedings of AAAI'10*, 121–127.

Lee, J. H. M., and Leung, K. L. 2012. Consistency Techniques for Flow-Based Projection-Safe Global Cost Functions in Weighted Constraint Satisfaction. *Journal of Artificial Intelligence Research* 43:257–292.

Parrello, B.; Kabat, W.; and Wos, L. 1986. Job-shop Scheduling using automated reasoning: A Case Study of the Car-Sequence problem. *Journal of Automated Reasoning* 2(1):1–42.

Pesant, G. 2004. A Regular Language Membership Constraint for Finite Sequences of Variables. In *Proceedings of CP'04*, 482–495.

Sanchez, M.; de Givry, S.; and Schiex, T. 2008. Mendelian Error Detection in Complex Pedigrees using Weighted Constraint Satisfaction Techniques. *Constraints* 13(1):130–154.

Schiex, T.; Fargier, H.; and Verfaillie, G. 1995. Valued Constraint Satisfaction Problems: Hard and Easy Problems. In *Proceedings of IJCAI'95*, 631–637.

Solnon, C.; Cung, V.; Nguyen, A.; and Artigues, C. 2008. The Car Sequencing Problem: Overview of State-of-the-Art Methods and Industrial Case-Study of the ROADDEF'2005 Challege Problem. *European Journal of Operational Research* 191(3):912–927.

van Hoeve, W.-J.; Pesant, G.; and Rousseau, L.-M. 2006. On Global Warming: Flow-based Soft Global Constraints. *J. Heuristics* 12(4-5):347–373.

Zytnicki, M.; Gaspin, C.; and Schiex, T. 2009. Bounds Arc Consistency for Weighted CSPs. *Journal of Artificial Intelligence Research* 35:593–621.