Propagating Polynomially (Integral) Linear Projection-Safe Global Cost Functions in WCSPs

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

***Abstract*—Lee and Shum consider cost functions that are Polynomially Linear Projection-Safe (PLPS), but whose minimum cost computation is usually NP-hard. They suggest how such cost functions can still be efficiently propagated using relaxed forms of common consistencies. In this paper, we show that conjunctions of PLPS cost functions are still PLPS, and Lee and Shum's relaxed consistency method is applicable to give better runtime behavior. We further introduce Polynomially Integral Linear Projection-Safe (PILPS) cost functions, a subclass of PLPS cost functions, which have (a) linear formulations with size polynomial to the number of variables and domain sizes, (b) optimal solutions of the linear relaxation always being integral and (c) the last two conditions unaffected by projections/extensions, even though the operations modify the structure of cost functions. We show that conjunctions of PILPS cost functions are PLPS, which still satisfy conditions (a) and (c). Given a standard WCSP consistency α, we give theorems showing that maintaining relaxed α on a conjunction of PILPS cost functions is stronger than maintaining α on the individual cost functions. A useful application of our method is on some PILPS global cost functions, whose minimum cost computations are tractable and yet those for their conjunctions are not. Experiments are conducted to confirm empirically that maintaining relaxed consistencies on the conjoined cost functions is orders of magnitude more efficient, both in runtime and search space reduction, than maintaining the corresponding standard consistencies on the individual cost functions.**

I. Introduction

Weighted Constraint Satisfaction Problems (WCSPs) provide a framework for modeling over-constrained and optimization problems. The basic solution technique for WCSPs is branch-and-bound search augmented with various forms of standard consistencies, such as NC* [1], (strong) ∅IC [2], [3], (G)AC* [1], [4], [3], FD(G)AC* [5], [3], and (weak) ED(G)AC* [6], [3]. Enforcement of these consistencies aims at inferring good lower bounds for the WCSP at hand, and relies on efficient minimum cost computation of the cost functions in the problem.

Global cost functions are important for the success of WCSP for their versatility in modeling real applications and efficient propagation algorithms. Flow-based projection-safe [3] and polynomially decomposable global cost functions [7] are ones with good structures allowing polytime algorithms for minimum cost computations and the nice structures are unaffected by the projection/extension operations. On the other hand, many global cost functions are useful, and yet either their minimum cost computations are NP-hard or no polytime algorithms are discovered yet. Lee and Shum [8], [9] introduce the notion of *Polynomially Linear Projection-Safety (PLPS)*. If a difficult cost function is PLPS, then a good lower bound for the minimum cost of the cost function can still be computed efficiently using linear programming techniques. Relaxed versions of the standard consistencies, such as GAC*, FDGAC* and weak EDGAC*, are defined using the approximated minimum costs.

In this paper, we consider conjunctions of global cost functions sharing more than one variable. We give theorems showing that propagating on a conjunction using the standard consistencies is stronger than propagating on the individual cost functions. Unfortunately, the same is not true if we propagate using the *relaxed* consistencies with linear programming. Nevertheless, we present empirical results to demonstrate the benefits of propagating on conjunctions both in terms of runtime and pruning in general.

We introduce and give sufficient conditions for a special subclass of PLPS cost functions, namely *polynomially integral linear projection-safe (PILPS)* cost functions. Every PLPS function has a corresponding integer linear program formulation. We require the integer linear program formulation of an PILPS cost function to have exactly the same solutions as its linear relaxation. An important consequence is that enforcing the standard consistencies on PILPS cost functions is exactly the same as enforcing their relaxed counterparts. In addition, the minimum cost of an PILPS function can be computed in polytime. The same is not necessarily true for conjunctions of PILPS cost functions, which we show to be still PLPS. Our central results show that propagating on individual PILPS cost functions using the standard (or relaxed since they are the same) consistencies is weaker than propagating on the conjunction of all these PILPS cost functions using the relaxed versions of the consistencies, which is in turn weaker than propagating on the conjunction using the standard consistency. The latter is NP-hard in general. These results give an exact characterization of the strength of the relaxed and standard consistencies on conjunctions of PILPS cost functions as compared against the corresponding standard consistencies on individual PILPS cost functions. Therefore, it is always more desirable to propagate on conjunctions of PILPS cost functions using even just relaxed consistencies.

Our central theorems are useful when we have cost functions

whose minimum cost computation is polytime but that for conjunctions of such cost functions is not. We show that flow-based projection safe [3] and an important subclass of polynomially decomposable [7] cost functions are PILPS. Although there exist polytime algorithms to compute the minimum costs of flow-based projection-safe and polynomially decomposable cost functions, we can prune more by propagating on conjunctions of such cost functions using just relaxed consistencies instead of propagating on individual cost functions using the corresponding standard consistencies. We conduct experiments to evaluate our method against the flow-based and polynomially decomposable approaches as well as pure integer programming, and observe orders of magnitude in runtime and search space improvements.

II. Background

A *Weighted Constraint Satisfaction Problem* (WCSP) [10] is a tuple ($\mathcal{X}$, $\mathcal{D}$, $\mathcal{C}$, k). $\mathcal{X}$ is a set of $variables$ $\{x_1,x_2,\ldots,x_n\}$. Each variable has its finite $domain$ $D(x_i) \in \mathcal{D}$ of values that can be assigned to it. Each variable can only be assigned with one value in its corresponding domain. An assignment on a set of variables can be represented by a tuple ℓ. We denote $\ell[x_i]$ the value assigned to x_i, $\ell[S]$ the tuple formed from the assignment on variables in the set $S \subseteq \mathcal{X}$, and $\mathcal{L}(S)$ a set of tuples corresponding to all possible assignments on the set of variables S. $\mathcal{C}$ is a set of *cost functions* W_S, each with scope S. W_S maps tuples $\mathcal{L}(S)$ to a cost valuation structure $V(k) = ([0 \ldots k], \oplus, \leq)$. The structure $V(k)$ contains a set of integers $[0, \ldots, k]$ with standard integer ordering $\leq$. Addition $\oplus$ is defined by $a \oplus b = \min(k, a + b)$. The subtraction $a \ominus b$ for $a, b \in [0 \ldots k]$ and $a \geq b$ is defined as $a \ominus b = a - b$ if $a \neq k$ and $k \ominus a = k$ for any a.

Without loss of generality, we assume $\mathcal{C} = \{W_\varnothing\} \cup \{W_i \mid x_i \in \mathcal{X}\} \cup \mathcal{C}^+$. $W_\varnothing$ is the constant nullary cost function, representing the lower bound of the WCSP. W_i is a unary cost function associated with variable $x_i \in \mathcal{X}$. C^+ is a set of cost functions with scopes of two or more variables.

The *cost* of a tuple ℓ for a WCSP corresponding to an assignment on $\mathcal{X}$ is defined as $cost(\ell) = W_\varnothing \oplus \bigoplus_{x_i \in \mathcal{X}} W_i(\ell[x_i]) \oplus \bigoplus_{W_S \in \mathcal{C}^+} W_S(\ell[S])$. A tuple ℓ is *feasible* if $cost(\ell) < k$. Our goal is to find a tuple ℓ which has the minimum cost among all the feasible tuples, and such a tuple is a *solution* of the WCSP. For convenience, we write $\min\{W_S\}$ to denote $\min\{W_S(\ell) \mid \ell \in \mathcal{L}(S)\}$.

WCSPs are typically solved with basic branch-and-bound search augmented with different consistency techniques. NC* [1], (strong) ∅IC [2], [3], AC* [1], FDAC* [5], and EDAC* [6] are defined for binary cost functions.

The enforcements of different consistencies involve finding the minimum costs of the cost functions, and moving those costs between cost functions by *projections* and *extensions* [11]. Projections move costs from n-nary cost functions to unary cost functions and from unary cost functions to the nullary one $W_\varnothing$. Given $S_2 \subset S_1$, a 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}), where

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

If $S_2 = \varnothing$, it is a projection to $W_\varnothing$. Extensions are the inverse of projections, and are defined similarly. In this paper, we restrict that $|S_2| \leq 1$. We assume that the minimum cost of the cost functions $\min\{W'_S\}$ cannot be smaller than 0 after a projection or extension operation.

A *global cost function* is a cost function with special semantics, based on which efficient algorithms can be designed for consistency enforcements. We denote a global cost function as SOFT_GC$^\mu(S)$ if it is derived from the corresponding hard *global constraint* GC with a *violation measure* μ and variable scope S. SOFT_GC$^\mu(S)$ returns 0 *iff* a given tuple ℓ on S satisfies GC. If ℓ violates GC, SOFT_GC$^\mu(S)$ returns $\mu(\ell)$ using the violation measure to reflect how much the GC is violated. An example is the SOFT_ALLDIFF^{var} [12] cost function derived from the ALLDIFF constraint, which restricts variables to take distinct values. The *variable-based* violation measure var returns the minimum number of variable assignments that needed to be changed for the constraint ALLDIFF to be satisfied.

Definition 1: Let $D_X = \bigcap_{i=1}^n D_{x_i}$, then

$$\begin{aligned} & \text{SOFT\_ALLDIFF}^{var}(x_1, \ldots, x_n) \\ = \; & \textstyle\sum_{d \in D_X} \max(|\{i \mid x_i = d\}| - 1, 0) \end{aligned}$$

To handle global cost functions which are usually of high-arity, standard consistencies are generalized to GAC* [4] and FDGAC* [3], and weak EDGAC* [3].

Lee and Leung [3] define $\mathcal{T}$ *projection-safety*. A cost function W_S is $\mathcal{T}$ *projection-safe* if (a) W_S satisfies property $\mathcal{T}$, and (b) W'_S satisfies property $\mathcal{T}$, where W'_S is obtained from W_S by a valid sequence of projections or extensions. In other words, the property $\mathcal{T}$ is preserved on W_S after projections and extensions.

Two useful properties $\mathcal{T}$ are flow-basedness and polynomially decomposable. *Flow-based projection-safe cost functions* [3] can be represented as flow networks, the minimum cost of which can be computed efficiently by flow algorithms. *Polynomially decomposable cost functions* [7] can be represented as dynamic programs, which allow the minimum costs to be computed efficiently using divide-and-conquer and memorization.

A cost function W_S is *linear* [8], [9] if it can be represented by an *integer linear program* [13] I_{W_S}, such that $\min\{W_S\}$ is equal to the minimum of I_{W_S}. W_S is *polynomially linear* if it is linear and the size of I_{W_S} is polynomial to $|S|$ and the maximum size of variable domains. W_S is *polynomially linear projection-safe (PLPS)* if it is polynomially linear and remains polynomially linear after projections and extensions.

Example 1: The SOFT_ALLDIFF$^{var}(x_1, \ldots, x_n)$ cost function is *PLPS* as it can be represented by the following

integer linear program I:

$$\min \sum_{a \in D_X} u_a, \quad s.t.$$

$$\sum_{i=1}^{n} c_{x_i,a} + u_a \leq 1, \forall a \in D_X$$

$$u_a \geq 0, \forall a \in D_X; c_{x_i,a} = \{0,1\}, \forall a \in D_X, 1 \leq i \leq n$$

The minimum of $I = \min\{\text{SOFT\_ALLDIFF}^{var}(x_1, \ldots, x_n)\}$. Thus $\text{SOFT\_ALLDIFF}^{var}(x_1, \ldots, x_n)$ is polynimally linear. As there exists a variable $c_{x_i,a}$ in I for every value $a \in D(x_i)$ in every variable x_i, I can be modified easily to represent $\text{SOFT\_ALLDIFF}^{var}(x_1, \ldots, x_n)$ after projections and extensions. So $\text{SOFT\_ALLDIFF}^{var}(x_1, \ldots, x_n)$ remains polynomially linear after projections and extensions and thus it is PLPS.

Solving integer linear programs can require exponential time in general, but a good approximation of the minimum cost $approx\_min\{W_S\}$ can be computed with the linear relaxation using linear programming. The approximated minimum cost forms the basis of relaxed but weaker forms of standard consistencies.

Standard consistencies for global cost functions include GAC*, FDGAC* and weak EDGAC*. An important concept for these consistencies are the notions of simple and full supports, which can be expressed (and enforced) in terms of the minimum cost $\min\{W_S\}$ of cost functions W_S. The relaxed form of these consistencies can be easily formulated by replacing $\min\{W_S\}$ by $approx\_min\{W_S\}$. We illustrate the idea using the definition of *relaxed GAC** as an example.

A variable x_i is *NC** [1] if (1) each value $v \in D(x_i)$ satisfies $W_i(v) \oplus W_\varnothing < k$ and (2) there exists a value $v' \in D(x_i)$ such that $W_i(v') = 0$. A WCSP is NC* *iff* all variables are NC*. A variable $x_i \in S$ is *GAC** [4] with respect to a cost function W_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 ℓ with $W_S(\ell) = 0$. $\{v_j\}$ 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. Note that the second requirement can be reformulated as follows.

- for each value $v_i \in D(x_i)$, $\min\{W_S(\ell) \,|\, l \in \mathcal{L}(S) \wedge \ell[x_i] = v_i\} = 0$.

By replacing "min" by "$approx\_min$" and "=" by "$\leq$", we get relaxed GAC*. A variable $x_i \in S$ is *relaxed GAC** [8], [9] with respect to a cost function W_S if:

- x_i is NC*, and;
- for each value $v_i \in D(x_i)$, $approx\_min\{W_S(\ell) \mid l \in \mathcal{L}(S) \wedge \ell[x_i] = v_i\} \leq 0$.

Note that the real number approximated minimum cost can be negative. The definitions of relaxed FDGAC* and relaxed weak EDGAC* can be constructed similarly.

III. Conjoining PLPS Cost Functions

If two constraints or cost functions share variables, they are *overlapping*. In the rest of the paper, we assume overlapping cost functions share more than one variable. In general, enforcing a consistency on the individual cost functions may not imply the same consistency on the conjunction of the two. An example is given by Bessière *et al.* [14] in which enforcing GAC on two overlapping ALLDIFF constraints does not imply GAC on the conjunction of them. It is easy to check that the result also holds for cost functions. By discovering extra pruning opportunities, propagating on conjunctions of cost functions may reduce more search space than propagating on individual cost functions can.

Every PLPS cost function has an associated integer linear program. PLPS cost functions can be conjoined together easily by combining their corresponding integer linear programs in a straightforward manner. Given two integer linear programs $I_{W_{S_1}}$ and $I_{W_{S_2}}$, we define $I_{W_{S_1}} \wedge I_{W_{S_2}}$ to be their combination by taking the union of their linear inequalities and adding up their objective functions. The following theorem ensures that conjunctions of PLPS cost functions remain PLPS.

Lemma 1: Suppose W_{S_1} and W_{S_2} are PLPS cost functions. The conjunction $W_{conj} \equiv W_{S_1} \wedge W_{S_2}$ is also PLPS.

Proof: Suppose W_{S_1} and W_{S_2} have their corresponding integer linear program $I_{W_{S_1}}$ and $I_{W_{S_2}}$ respectively. The integer linear program $I_{W_{conj}}$ for W_{conj} can simply be formed by $I_{W_{conj}} \equiv I_{W_{S_1}} \wedge I_{W_{S_2}}$. It is easy to check that W_{conj} satisfies the conditions for PLPS. ■

An immediate question is whether a conjunction of PLPS cost functions always gives a stronger bound than using the individual PLPS cost functions, given that the same level of consistency is maintained. Given WCSP $P_{PLPS} = (\mathcal{X}, \mathcal{D}, \mathcal{C}_{PLPS}, k)$, where each cost function $W_S \in \mathcal{C}_{PLPS}$ is PLPS with corresponding integer linear program I_{W_S}. We assume that $\mathcal{C}_{PLPS}$ contains overlapping cost functions. We can construct an equivalent WCSP $P_{conj} = (\mathcal{X}, \mathcal{D}, \mathcal{C}_{conj}, k)$ where $\mathcal{C}_{conj} = \{W_{conj}\}$ and $W_{conj} \equiv \bigwedge_{W_S \in \mathcal{C}_{PLPS}} W_S$ with the corresponding scope $S_{conj} \equiv \bigcup_{W_S \in \mathcal{C}_{PLPS}} S$ and integer linear program $I_{W_{conj}} \equiv \bigwedge_{W_S \in \mathcal{C}_{PLPS}} I_{W_S}$. Since $\mathcal{C}_{PLPS}$ is a set of PLPS cost functions, the conjunction W_{conj} must be an PLPS cost function.

Given a problem P representable by two WCSP models $\phi(P)$ and $\psi(P)$. A consistency Φ on $\phi(P)$ is *strictly stronger* than another consistency Ψ on $\psi(P)$, written as Φ on $\phi(P) > \Psi$ on $\psi(P)$, iff $\psi(P)$ is Ψ whenever $\phi(P)$ is Φ, but not *vice versa* [3].

We show that (FD)GAC* and weak EDGAC* on P_{conj} are strictly stronger than their counterparts on P_{PLPS} respectively by the following theorem.

Theorem 1: Suppose α-consistency is one of GAC*, FDGAC*, and weak EDGAC*. We have α-consistent on $P_{conj} > \alpha$-consistent on P_{PLPS}

Proof: We prove the part for GAC*. The proofs for the other consistencies are similar.

Assume P_{conj} is GAC*, but P_{PLPS} is not GAC*. There exists a variable $x_i \in \mathcal{X}$ with a value $a \in D(x_i)$ and a cost func-

tion $W_S \in \mathcal{C}_{PLPS}$ in P_{PLPS} such that $\min\{W_S(\ell) \mid \ell[x_i] = a \wedge \ell \in \mathcal{L}(S)\} > 0$. Now, we have

$$\begin{array}{rl} & \min\{W_{conj} \mid \ell[x_i] = a \wedge \ell \in \mathcal{L}(S_{conj})\} \\ \geq & \bigoplus_{W_S \in \mathcal{C}_{PLPS}} \min\{W_S(\ell) \mid \ell[x_i] = a \wedge \ell \in \mathcal{L}(S)\} > 0 \end{array}$$

So we cannot find a simple support for a and x_i cannot be GAC* with respect to W_{conj} in P_{conj}.

Consider $W_{S_1} = \text{SOFT\_ALLDIFF}^{var}(x_1, x_2, x_3)$ and $W_{S_2} = \text{SOFT\_ALLDIFF}^{var}(x_2, x_3, x_4)$, where $D(x_1) = \{a, b\}$, $D(x_2) = D(x_3) = \{a, b, c\}$ and $D(x_4) = \{b, c\}$. It is easy to check that $P_{PLPS} = (\mathcal{X}, \mathcal{D}, \{W_{S_1}, W_{S_2}\}, k)$ is GAC*. However, $P_{conj} = (\mathcal{X}, \mathcal{D}, \{W_{S_1} \wedge W_{S_2}\}, k)$ is not GAC* since the minimum cost when $x_1 = a$ is $1 > 0$.

Result follows. ■

When standard consistencies are replaced by their relaxed versions, result similar to that of Theorem 1 does not hold. For simplicity, we assume $\mathcal{C}_{PLPS} = \{W_{S_1}, W_{S_2}\}$. Suppose P_{conj} is relaxed GAC*. We have

$$\begin{array}{rl} 0 & \geq \text{approx\_}\min\{W_{conj} \mid \ell[x_i] = a \wedge \ell \in \mathcal{L}(S_{conj})\} \\ & \geq \bigoplus_{W_S \in \mathcal{C}_{PLPS}} \text{approx\_}\min\{W_S(\ell) \mid \ell[x_i] = a \wedge \ell \in \mathcal{L}(S)\} \end{array}$$

In order for the sum to be non-positive, it is possible for the approximated minimum cost of one of $\{W_{S_1}, W_{S_2}\}$ to be negative and the other one positive. Therefore, one of them is not relaxed GAC*.

The peculiar bad situation just described does not happen often in practice. We demonstrate that it is worthwhile to propagate on the conjunction instead of individual cost functions. We use the same setup as the experiments by Lee and Shum [8] and compare against their experimental results using the same set of problem instances, which includes the generalized version of the soft car sequencing problem, the soft magic square problem and the weighted tardiness scheduling problem. In our models, all PLPS cost functions in a problem are conjoined into a single one.

Results are shown in Tables 1, 2, and 3. The average number of backtrack (bt) and the average runtime in seconds (time) for solved cases are reported. The runtime includes the CPU time used by both the WCSP solver Toulbar2 and the linear program solver CPLEX. Next to the runtime, we also report separately in brackets the CPU time used by the linear programing solver (CPLEX) with a "*" if the execution of one of the 10 instances exceeds the timeout of 3600 seconds. The best result among those with the most cases solved is highlighted in bold. The models using the conjunction of PLPS cost functions run faster in instances with larger size, and prune more in all cases. Greater improvements are shown in harder instances, e.g. the largest instance in Table III. Since the instances only contain PLPS cost functions, they are conjoined into a single PLPS cost function in our model. In this model, there is no propagation between cost functions and the effects of relaxed GAC*, relaxed FDGAC*, and relaxed weak EDGAC* are similar. So relaxed FDGAC* and relaxed weak EDGAC* do not infer a much better bound than relaxed GAC* when conjoined PLPS cost functions are used. The reduction in search space does not compensate for the pruning overhead, and the simpler and less costly relaxed GAC* gives the best results in terms of run-time.

TABLE I
THE GENERALIZED SOFT CAR SEQUENCING PROBLEM USING SOFT_SLIDINGSUM^{dec}

Modeling with PLPS cost functions						
n	relaxed GAC*		relaxed FDGAC*		relaxed weak EDGAC*	
	bt	time (CPLEX)	bt	time (CPLEX)	bt	time (CPLEX)
8	19.0	0.21 (0.20)	9.2	0.20 (0.19)	**9.0**	0.25 (0.23)
10	41.2	0.55 (0.51)	21.0	0.52 (0.49)	19.8	0.73 (0.68)
12	119.6	1.44 (1.35)	48.2	1.15 (1.07)	45.6	1.34 (1.26)
14	585.1	17.63 (16.91)	264.8	13.12 (12.76)	249.0	15.19 (14.61)
Modeling with conjoined PLPS cost function						
n	relaxed GAC*		relaxed FDGAC*		relaxed weak EDGAC*	
	bt	time (CPLEX)	bt	time (CPLEX)	bt	time (CPLEX)
8	16.0	**0.19 (0.18)**	12.4	0.30 (0.28)	12.4	0.38 (0.35)
10	30.6	**0.46 (0.43)**	20.0	0.71 (0.65)	**17.8**	0.86 (0.80)
12	86.4	**1.07 (1.01)**	43.0	1.52 (1.45)	**36.4**	1.62 (1.55)
14	133.0	**1.30 (1.26)**	74.2	1.77 (1.71)	**64.1**	1.77 (1.72)

TABLE II
THE SOFT MAGIC SQUARE PROBLEM USING SOFT_EGCC^{var}

Modeling with PLPS cost functions						
n	relaxed GAC*		relaxed FDGAC*		relaxed weak EDGAC*	
	bt	time (CPLEX)	bt	time (CPLEX)	bt	time (CPLEX)
9	23.1	**0.24 (0.23)**	19.3	0.27 (0.26)	17.1	0.43 (0.41)
12	54.7	**0.71 (0.65)**	44.9	0.99 (0.93)	42.3	1.52 (1.43)
15	89.2	1.70 (1.59)	53.1	2.32 (2.21)	50.2	3.64 (3.46)
18	93.7	3.03 (2.89)	64.7	4.80 (4.64)	59.8	6.41 (6.13)
Modeling with conjoined PLPS cost function						
n	relaxed GAC*		relaxed FDGAC*		relaxed weak EDGAC*	
	bt	time (CPLEX)	bt	time (CPLEX)	bt	time (CPLEX)
9	12.8	0.27 (0.25)	9.8	0.43 (0.40)	**9.7**	0.54 (0.50)
12	33.5	0.86 (0.81)	24.6	1.55 (1.48)	**24.3**	1.89 (1.80)
15	39.2	**1.62 (1.52)**	32.6	2.95 (2.86)	**32.3**	3.64 (3.51)
18	49.4	**2.96 (2.87)**	36.7	5.78 (5.48)	**36.4**	7.29 (6.82)

TABLE III
THE WEIGHTED TARDINESS SCHEDULING PROBLEM USING SOFT_DISJUNCTIVE^{val}

Modeling with PLPS cost functions						
$n, d, \lvert T\rvert$	relaxed GAC*		relaxed FDGAC*		relaxed weak EDGAC*	
	bt	time (CPLEX)	bt	time (CPLEX)	bt	time (CPLEX)
3,3,12	7.0	**0.05 (0.05)**	**6.0**	0.06 (0.06)	**6.0**	0.08 (0.08)
4,4,20	13.0	**0.14 (0.13)**	**8.0**	0.18 (0.17)	**8.0**	0.25 (0.24)
5,5,30	35.0	0.60 (0.56)	19.0	0.68 (0.63)	**15.0**	0.98 (0.92)
6,5,35	382.0	7.01 (6.75)	32.1	1.90 (1.82)	28.1	2.41 (2.32)
7,5,40	2253.6	61.89 (60.14)	27.0	2.78 (2.47)	25.2	3.51 (3.32)
8,5,45	*	* (*)	214.0	22.09 (21.23)	210.1	30.16 (28.90)
Modeling with conjoined PLPS cost function						
$n, d, \lvert T\rvert$	relaxed GAC*		relaxed FDGAC*		relaxed weak EDGAC*	
	bt	time (CPLEX)	bt	time (CPLEX)	bt	time (CPLEX)
3,3,12	**6.0**	**0.05 (0.05)**	**6.0**	0.09 (0.09)	**6.0**	0.16 (0.13)
4,4,20	**8.0**	0.15 (0.14)	**8.0**	0.29 (0.28)	**8.0**	0.35 (0.34)
5,5,30	15.5	**0.53 (0.50)**	15.2	1.04 (1.01)	**15.0**	1.29 (1.25)
6,5,35	23.2	**0.95 (0.90)**	18.8	1.90 (1.83)	**18.0**	2.32 (2.24)
7,5,40	35.2	**1.72 (1.64)**	27.0	3.44 (3.30)	**21.0**	4.23 (4.07)
8,5,45	40.1	**3.49 (3.24)**	34.5	7.38 (7.08)	**33.1**	8.71 (8.31)

IV. CONJOINING TRACTABLE GLOBAL COST FUNCTIONS OTHER THAN PLPS

Polynomially Integral Linear Projection-Safe (PILPS) cost functions form a special subclass of PLPS cost functions [8], [9]. A cost function W_S is *polynomially integral linear* if W_S is (a) polynomially linear and (b) the optimal solution, if it exists, of the linear relaxation of its corresponding integer linear program I_{W_S} is always integral.

Lemma 2: Polynomially integral linear cost functions are polynomially linear.

An immediate observation is that the exact minimum cost of a polynomially integral linear cost function can be obtained by solving the linear relaxation of their corresponding integer linear programs.

Lemma 3: If W_S is a polynomially integral linear cost function, $\min\{W_S\} = \text{approx\_}\min\{W_S\}$.

Theorem 2: Minimum cost computation of polynomially integral linear cost functions is polynomial.

Proof: Since $\min\{W_S\} = \text{approx\_}\min\{W_S\}$, $\min\{W_S\}$ can be determined using interior point algorithms [13] for linear programs with the worst case complexity bounded by polynomial time. ■

Recall the notion of $\mathcal{T}$ projection-safety. In addition to flow-basedness and polynomially linearity, polynomially integral linearity is another good property $\mathcal{T}$ to be maintained across projections/extensions. Therefore, it makes sense to require cost functions to be *polynomially integral linear projection-safe (PILPS)*.

We give a possible sufficient condition to identify PILPS cost functions.

Theorem 3: A cost function W_S is *polynomially integral linear projection-safe* if:

1) W_S is linear and has the corresponding integer linear program I_{W_S}, where the number of inequalities and number of variables of I_{W_S} are polynomial to the number of variables and the maximum domain size of W_S;
2) there exists a surjective function Λ' mapping each optimal solution $\gamma_{I_{W_S}}$ in I_{W_S} to each tuple $\ell[S] \in L(S)$, where $L(S)$ denotes the set of tuples corresponding to all possible assignments on variables S, and;
3) for each value $v \in D(x_i)$ in each variable $x_i \in S$, there exists an injective function mapping an assignment $\{x_i \mapsto v\}$ to a 0-1 variable $c_{x_i,v}$ in I_{W_S} such that if $\ell[S] = \Lambda'(\gamma_{I_{W_S}})$ for an optimal solution $\gamma_{I_{W_S}}$ in I_{W_S} and a tuple $\ell[S] \in L(S)$, whenever $\ell[x_i] = v$ for some tuple $\ell[S]$, $\gamma_{I_{W_S}}[c_{x_i,v}] = 1$; whenever $\ell[x_i] \neq v$, $\gamma_{I_{W_S}}[c_{x_i,v}] = 0$
4) I_{W_S} is *totally dual integral* or the associated matrix of I_{W_S} is *totally unimodular*.

Proof: If a linear program is *totally dual integral* or its associated matrix is *totally unimodular*, its optimal solutions must be integral [15]. By augmenting conditions 1, 2, and 3 from the sufficient condition for PLPS cost functions [8], [9] straightforwardly, the sufficient conditions for PILPS is formed as given. ■

Example 2: The SOFT_ALLDIFF$^{var}(x_1, \ldots, x_n)$ cost function is *PILPS* as the associated matrix of its corresponding integer linear program given in Example 1 is totally unimodular.

Polynomially integral linear projection-safe cost functions are interesting since their conjunctions are PLPS.

By Lemma 1 and 2, we have the following corollary.

Corollary 1: Suppose W_{S_1} and W_{S_2} are PILPS cost functions. The conjunction $W_{conj} \equiv W_{S_1} \wedge W_{S_2}$ is PLPS.

Corollary 2: Suppose W_S is PILPS, and α-consistency is one of GAC*, FDGAC* and weak EDGAC*. Relaxed α-consistent on W_S is equivalent to α-consistent on W_S.

In general, it is NP-hard to compute the minimum cost of the conjunction of overlapping PILPS cost functions, since the conjunction of their corresponding linear programs may not always give integral minimums when there exists a minimum [13]. As the conjunction of PILPS cost functions remains PLPS, linear programming techniques allow its approximated minimum cost to be computed efficiently, and relaxed form of standard consistencies can thus be enforced. We have the following result when relaxed consistencies are enforced on the conjunction of PILPS cost functions compared to the corresponding (non-relaxed) consistencies enforced on the individual cost functions.

Given WCSP $P_{PILPS} = (\mathcal{X}, \mathcal{D}, \mathcal{C}_{PILPS}, k)$, where each cost function $W_S \in \mathcal{C}_{PILPS}$ is PILPS with corresponding integer linear program I_{W_S}. We assume that $\mathcal{C}_{PILPS}$ contains overlapping cost functions. We can construct an equivalent WCSP $P_{conj} = (\mathcal{X}, \mathcal{D}, \mathcal{C}_{conj}, k)$ where $\mathcal{C}_{conj} = \{W_{conj}\}$ and $W_{conj} \equiv \bigwedge_{W_S \in \mathcal{C}_{PILPS}} W_S$ with the corresponding scope $S_{conj} \equiv \bigcup_{W_S \in \mathcal{C}_{PILPS}} S$ and integer linear program $I_{W_{conj}} \equiv \bigwedge_{W_S \in \mathcal{C}_{PILPS}} I_{W_S}$.

We show that relaxed (FD)GAC* and relaxed weak EDGAC* on P_{conj} are strictly stronger than (FD)GAC* and weak EDGAC* on P_{PILPS} respectively by the following theorem.

Theorem 4: Suppose α-consistency is one of GAC*, FDGAC* and weak EDGAC*. Relaxed α-consistent on P_{conj} is strictly stronger than α-consistent on P_{PILPS}.

Proof: We prove the part for relaxed GAC*. The proofs for the other consistencies are similar.

Assume P_{conj} is relaxed GAC*, but P_{PILPS} is not GAC*. There exists a variable $x_i \in \mathcal{X}$ with a value $a \in D(x_i)$ and a cost function $W_S \in \mathcal{C}_{PILPS}$ in P_{PILPS} such that $\min\{W_S(\ell) \,|\, \ell[x_i] = a \wedge \ell \in \mathcal{L}(S)\} > 0$. Since all cost functions $W_S \in \mathcal{C}_{PILPS}$ are PILPS, we have

$$\begin{aligned} & \text{approx\_}\min\{W_{conj} \mid \ell[x_i] = a \wedge \ell \in \mathcal{L}(S)\} \\ \geq\ & \bigoplus_{W_S \in \mathcal{C}_{PILPS}} \text{approx\_}\min\{W_S \mid \ell[x_i] = a \wedge \ell \in \mathcal{L}(S)\} \\ =\ & \bigoplus_{W_S \in \mathcal{C}_{PILPS}} \min\{W_S(\ell) \mid \ell[x_i] = a \wedge \ell \in \mathcal{L}(S)\} > 0 \end{aligned}$$

Thus, a cannot have any simple support and x_i cannot be relaxed GAC* with respect to W_{conj} in P_{conj}.

Consider $W_{S_1} = \text{SOFT\_ALLDIFF}^{var}(x_1, x_2, x_3)$ and $W_{S_2} = \text{SOFT\_ALLDIFF}^{var}(x_2, x_3, x_4)$, where $D(x_1) = \{a, b\}$, $D(x_2) = D(x_3) = \{a, b, c\}$ and $D(x_4) = \{b, c\}$. It is easy to check that $P_{PILPS} = (\mathcal{X}, \mathcal{D}, \{W_{S_1}, W_{S_2}\}, k)$ is GAC*. However, $P_{conj} = (\mathcal{X}, \mathcal{D}, \{W_{S_1} \wedge W_{S_2}\}, k)$ is not relaxed GAC* since the approximated minimum cost when $x_1 = a$ is $1 > 0$.

Result follows. ■

Since relaxed consistencies are the weaker forms of standard consistencies, we have the following lemma.

Lemma 4: Suppose α-consistency is one of GAC*, FDGAC* and weak EDGAC*. We have α-consistent on P_{conj} $>$ relaxed α-consistent on P_{conj}.

Enforcing α-consistency on P_{conj} infers better bounds, but it may not be worthwhile since it can be NP-hard. In contrast, the relaxed consistencies can be enforced efficiently on P_{conj}.

An immediate application of Theorem 4 is to existing global cost functions with polytime minimum cost computation. In many cases the minimum cost computation for their conjunctions is NP-hard. Theorem 4 suggest that it is still worthwhile to enforce relaxed consistencies on these cost functions. Flow-based projection-safe cost functions [3] and polynomially de-

composable cost functions [7] are such examples. By enforcing relaxed consistencies on their conjunctions, the search benefits from the better bounds inferred.

Theorem 5: Flow-based projection-safe cost functions are PILPS.

Proof: Every flow-based projection-safe cost function has a corresponding network flow problem, which in turn has a corresponding integer linear program with a totally unimodular matrix [15]. The cost function, the flow problem, and the integer linear program shares the same minimum cost. Since the integer linear program always has integral solutions when solved by linear relaxation, the result follows. ■

Corollary 3: The flow-based projection-safe cost functions [3] SOFT_ALLDIFF^{var}, SOFT_ALLDIFF^{dec}, SOFT_GCC^{var}, SOFT_GCC^{val}, SOFT_SAME^{var}, SOFT_SAME^{val}, SOFT_REGULAR^{var}, and SOFT_REGULAR^{edit} are PILPS cost functions.

We give the related definitions for polynomially decomposable cost functions to show that they are PILPS cost functions.

Lemma 5: [7] 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:

1) $\sum_{i=1}^{r} n_i$ is polynomial to $|S|$ and d, and;
2) 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.

Theorem 6: Suppose W_S is a polynomially decomposable cost function stated in Lemma 5, then W_S is PILPS.

Proof: We show that W_S is PILPS by first showing that it is flow-based projection-safe. The function stated in Lemma 5 consists of the operations $\min$ and $\bigoplus$, which can be represented and computed in flow networks. So, W_S can be represented as a min-cost flow problem with a corresponding flow network, where each cost function $\omega_{S_{i,j}}$ is represented by a node. The operation $\min$ is represented by connecting all the nodes of the related cost functions to the sink. The operation $\bigoplus$ is represented by a path connecting all the nodes of the related cost functions.

As a result, W_S is a flow-based projection-safe cost function. By Theorem 5, W_S is an PILPS cost functions. ■

Corollary 4: The polynomially decomposable cost functions [7] SOFT_AMONG^{var}, SOFT_REGULAR^{var}, SOFT_GRAMMAR^{var}, MAX_WEIGHT, and MIN_WEIGHT are PILPS.

We note that, for the cost functions mentioned above, their dedicated polynomial time algorithms are usually more efficient than interior point algorithms or linear programming.

By propagating the conjunction of cost functions, extra pruning opportunities can be discovered which may reduce more search space than propagating the individual cost functions. Unfortunately, it can be NP-hard to compute the minimum cost for the conjunction of PILPS cost functions even an efficient polynomial time algorithm is given for the individual cost functions.

Bessière *et al.* [14] show the above result on the hard ALLDIFF constraints and it can be generalized to the SOFT_ALLDIFF^{var}, SOFT_ALLDIFF^{dec}, SOFT_GCC^{var}, SOFT_GCC^{val}, and SOFT_SAME^{var} cost functions. Régin [16] also shows the above result on the hard AMONG [17] constraints, where an AMONG constraint restricts the number of variables to be assigned to a value from a specific set. The result can be generalized to the SOFT_AMONG^{var}, SOFT_REGULAR^{var}, SOFT_REGULAR^{edit}, and SOFT_GRAMMAR^{var} cost functions. Theorem 4 suggests that enforcing the relaxed consistencies on the conjunction of such PILPS cost functions can still be more efficient and worthwhile than handling them individually.

Given WCSP $P_{PILPS} = (\mathcal{X}, \mathcal{D}, \mathcal{C}_{PILPS}, k)$, where $\mathcal{C}_{PILPS}$ consists of some PILPS cost functions, and an equivalent WCSP $P_{conj} = (\mathcal{X}, \mathcal{D}, \mathcal{C}_{conj}, k)$ where $\mathcal{C}_{conj} = \{W_{conj}\}$ and $W_{conj} \equiv \bigwedge_{W_S \in \mathcal{C}_{PILPS}} W_S$. We give an example similar to the one given by Bessière *et al.* [14] in the following theorem. By propagating on a conjunction of PILPS cost functions with relaxed consistencies, a higher bound can be inferred earlier in an exponentially number of steps during the branch-and-bound search.

Theorem 7: Suppose α-consistency is one of GAC*, FDGAC* and weak EDGAC*. There exists a class of WCSP P_{PILPS}, so that if we enforce α-consistency on P_{conj} and α-consistency on P_{PILPS} in branch-and-bound search, an exponential search tree needs to be explored for P_{PILPS} to infer the same minimum cost as in the case of P_{conj}.

Proof: We prove the part for relaxed GAC*. The proofs for the other consistencies are similar. Given a WCSP $P_{PILPS} = (X \cup Y \cup Z, \mathcal{D}, \mathcal{C}_{PILPS}, k)$ where $X = \{x_1, \ldots, x_n\}$, $Y = \{x_1, \ldots, x_{2n}\}$, $Z = \{x_1, \ldots, x_n\}$, $\mathcal{C}_{PILPS} = \{$SOFT_ALLDIFF$^{var}(X \cup Y)$, SOFT_ALLDIFF$^{var}(Y \cup Z)\}$, $D(X_i) = [1, 2n-1]$, $i = 1, \ldots, n$, $D(Y_i) = [1, 4n-1]$, $i = 1, \ldots, 2n$, and $D(Z_i) = [2n, 4n-1]$, $i = 1, \ldots, n$.

Consider the WCSP $P_{conj} = (X \cup Y \cup Z, \mathcal{D}, \mathcal{C}_{conj}, k)$ where $\mathcal{C}_{conj} = \{W_{conj}\}$ and $W_{conj} \equiv$ SOFT_ALLDIFF$^{var}(X \cup Y) \wedge$ SOFT_ALLDIFF$^{var}(Y \cup Z)$. W_{conj} gives an approximated minimum cost approx_$\min\{W_{conj}\}$ of 1 which can be inferred by enforcing relaxed GAC* on $\mathcal{C}_{conj}$. On the other hand, a subset of n or fewer variables has at least $2n-1$ values in their domains and a subset of $n+1$ to $3n$ variables has $4n-1$ values in their domains. Thus, to infer a minimum cost of 1 in P_{PILPS} by enforcing GAC* on $\mathcal{C}_{PILPS}$, we must instantiate at least $n-1$ variables. ■

In addition to the theoretical results, we conduct experiments to show the efficiency of modeling cost functions as PILPS cost functions and propagating their conjunctions in the next section.

V. EXPERIMENTAL RESULTS

To demonstrate the efficiency of our framework, we compare the performances of (a) models using conjunctions of PILPS cost functions against (b) models using individual flow-based projection-safe / polynomially decomposable cost functions. The consistencies GAC*, FDGAC*, weak EDGAC*

and their relaxed versions are implemented in Toulbar2[1] v0.9. IBM ILOG CPLEX Optimizer[2] 12.2 is called from Toulbar2 to solve (integer) linear programs. Our benchmarks' models consist of both PILPS global cost functions as well as table cost functions, the latter of which are handled individually using exact minimum costs even when relaxed consistencies are used.

Variables with smaller domains and values with lower unary costs are assigned first. The experiments are conducted on an Intel Core2 Duo E7400 (2 x 2.80GHz) machine with 4GB RAM. In each benchmark we use different parameter settings to construct different instances, and 10 random cases are generated with each parameter setting. We use the timeout of 3600 seconds and report the average number of backtracks (bt) and the average runtime in seconds (time) for solved cases. The runtime includes the CPU time used by both the WCSP solver Toulbar2 and the linear programming solver CPLEX. Next to the runtime, we also report separately in brackets the CPU time used by the linear programming solver (CPLEX). We truncate the floating point variables in CPLEX at the 10-th decimal place. We mark the entries with a "*" if the execution of one of the 10 instances exceeds the timeout. The best result among those with the most cases solved is highlighted in bold.

The following benchmarks utilize the global cost functions described in the paper.

- The car sequencing problem (prob001 in CSPLib[3]) finds a sequence of n cars of $u \in U$ different types to be built. There is a set of options I which each type may or may not be equipped with, and each assembly line of an option $i \in I$ restricts that at most m_i cars for every s_i cars with that option equipped can be built. A GCC [18] constraint is used to ensure that the number of cars of each type is built according to the plan. Overlapping SOFT_AMONG$^{var}()$ [19] cost functions are used to ensure the restrictions of each assembly line are satisfied, and they are modeled by either polynomially decomposable cost functions or PILPS cost functions. There are preferences for each assembly line, e.g. two consecutive cars of the same type are preferred, and they are modeled by table cost functions. We fix $|I| = 5$ and $u = 5$ and use instances with different n in our experiments.
- The examination timetabling problem finds a schedule for n examinations over d days for s groups of students, Each group of students attends a set of at most d examinations and the number of days with more than 1 examination should be minimized for every group of students. A SOFT_ALLDIFF$^{var}()$ [12] cost function is used for every group of students, and they are modeled by either flow-based projection-safe cost functions or PILPS cost functions. There are preferences between examinations, e.g. the locations of two examinations are far away and should not be scheduled on the same day in case there are students attending both of them, and they are modeled by table cost functions. We fix $s = 4$ and use different n and d in our experiments.

We soften the above problems by replacing the global constraints by their soft variants, by either the flow-based projection-safe / polynomially decomposable implementations or the PILPS implementations. For each variable x_i introduced, a random unary cost from 0 to 9 is assigned to each value in $D(x_i)$. Random preferences are added to the instances in the form of table cost functions. The softened fair scheduling problem (in Global Constraint Catalog[4]) is also used in our experiments and similar results can be observed. We omit the details due to the paper space limitations.

Results are shown in Tables 4 and 5. In these benchmarks, models using conjunctions of PILPS cost functions using relaxed α-consistency run faster and prune more than models with individual flow-based projection-safe / polynomially decomposable cost functions using α-consistency in many cases, especially when the problem size is large. Note that models using PILPS cost functions contain also table cost functions, and are thus applied with a mix of relaxed α-consistency (for PILPS functions) and α-consistency (for table functions).

Stronger consistencies always have higher overhead. We gain in runtime only when the extra pruning can compensate for the overhead. This is not the case in general for relaxed weak EDGAC* in our easy problem instances as reported in these tables. That is why relaxed FDGAC* exhibits better runtime behavior than weak EDGAC*.

We use slightly easier problem instances so that we can make sensible comparisons with the weaker consistencies and the flow-based projection-safe / polynomially decomposable cost function implementations. Note that integer linear programming solver can also solve our benchmarks competitively. We use more difficult instances with more preferences (table cost functions) to compare the performances of modeling the problem with integer linear programs solved by the IBM ILOG CPLEX Optimizer 12.2 with both of the models above. We use the encoding method introduced by Koster [20] to formulate binary cost functions as integer linear programs. We only show the results for the models with flow-based projection-safe / polynomially decomposable (p.d.) cost functions using weak EDGAC* and PILPS cost functions using relaxed weak EDGAC* as those models have the best results among the other (relaxed) consistencies in the same model in this setting.

Results are shown in Tables 6 and 7. In almost all cases, our models using conjunctions of PILPS cost functions run faster and prune more. The trend is more apparent when the problem size grows.

VI. CONCLUSION

Global cost functions are indispensable in modeling complex real-life problems. The work reported here continues the quest for efficient implementations of such functions in WCSP

[1] http://carlit.toulouse.inra.fr/cgi-bin/awki.cgi/ToolBarIntro

[2] http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/

[3] http://www.csplib.org/

[4] http://www.emn.fr/x-info/sdemasse/gccat/

TABLE IV
THE SOFT CAR SEQUENCING PROBLEM

Modeling with the conjunction of PILPS cost functions						
n	relaxed GAC*		relaxed FDGAC*		relaxed weak EDGAC*	
	bt	time (CPLEX)	bt	time (CPLEX)	bt	time (CPLEX)
12	72.6	2.22 (1.58)	12.1	**0.47 (0.33)**	**12.0**	0.85 (0.67)
14	85.7	3.11 (2.39)	15.8	**0.73 (0.55)**	**15.0**	1.30 (1.07)
16	89.3	4.33 (3.47)	16.1	**1.13 (0.87)**	**15.6**	1.92 (1.59)
18	123.3	7.20 (5.87)	18.9	**1.38 (1.06)**	**18.0**	2.24 (1.89)
20	139.7	10.29 (8.51)	22.0	**2.01 (1.49)**	**20.6**	3.31 (2.69)
Modeling with polynomially decomposable cost functions						
n	GAC*		FDGAC*		weak EDGAC*	
	bt	time	bt	time	bt	time
12	23667.9	23.03	563.4	2.67	210.3	1.54
14	310845	328.49	2774.9	16.53	983.1	11.89
16	*	*	6653.2	53.06	2191.3	25.10
18	*	*	8104.2	93.87	3651.7	49.62
20	*	*	21285.5	303.10	8025.6	161.82

TABLE V
THE SOFT EXAMINATION TIMETABLING PROBLEM

Modeling with the conjunction of PILPS cost functions						
n, d	relaxed GAC*		relaxed FDGAC*		relaxed weak EDGAC*	
	bt	time (CPLEX)	bt	time (CPLEX)	bt	time (CPLEX)
25, 8	5507.6	405.67 (379.50)	29.8	3.77 (3.25)	**25.4**	4.32 (3.66)
30, 8	*	*	63.4	18.37 (16.09)	**50.4**	8.50 (6.99)
35, 8	*	*	35.8	7.96 (5.62)	**35.0**	10.85 (7.21)
30, 12	*	*	140.1	**26.49 (20.64)**	**124.7**	30.88 (22.91)
35, 12	*	*	93.0	**45.02 (37.15)**	**78.3**	51.41 (40.31)
Modeling with flow-based projection-safe cost functions						
n, d	GAC*		FDGAC*		weak EDGAC*	
	bt	time	bt	time	bt	time
25, 8	16747.8	41.514	97.8	**0.67**	92.6	0.68
30, 8	*	*	224.0	**7.93**	208.4	8.75
35, 8	*	*	72.2	0.51	62.4	**0.44**
30, 12	*	*	*	*	*	*
35, 12	*	*	*	*	*	*

TABLE VI
COMPARISON WITH ILP: SOFT CAR SEQUENCING

n	p.d. & weak EDGAC*		PILPS & relaxed weak EDGAC*		ILP
	bt	time	bt	time (CPLEX)	time
12	527.8	119.96	**37.8**	103.26 (68.73)	**63.28**
14	2287.2	788.94	**42.6**	**155.21 (135.49)**	177.79
16	6835.1	1828.22	**96.3**	**207.07 (175.64)**	386.30
18	*	*	**110.1**	**653.82 (549.44)**	662.56
20	*	*	**311.2**	**1163.03 (1026.89)**	1442.44

TABLE VII
COMPARISON WITH ILP: SOFT EXAMINATION TIMETABLING

n, d	flow-based & weak EDGAC*		PILPS & relaxed weak EDGAC*		ILP
	bt	time	bt	time (CPLEX)	time
25, 8	211.0	2.93	**47.0**	5.87 (4.22)	**2.29**
30, 8	1140.1	31.28	**105.0**	11.53 (9.61)	**10.76**
35, 8	704.2	19.77	**84.1**	**11.07 (8.17)**	12.56
30, 12	*	*	**790.1**	**544.01 (449.89)**	725.54
35, 12	*	*	**681.0**	**738.09 (640.58)**	876.47

solvers. Previous work suggests PLPS cost functions for inherently computationally expensive cost functions, and relaxed consistencies to handle them efficiently. The conjunction of global cost functions is well-known to give better bounds on different consistencies. Our work gives a practical way of conjoining PLPS cost functions by linear programming techniques. We show a subclass of PLPS cost functions gives better bounds on conjunctions with approximations compared to individual cost functions. While such individual cost functions are tractable to handle, their conjunctions can become intractable to manage. Our work suggests approximation is a useful technique to tackle conjunctions of such cost functions, and we demonstrate that it is beneficial both in terms of runtime and pruning.

Cooper *et al.* [21] suggest optimal soft arc consistency (OSAC), the implementation of which also utilizes linear programming techniques, for WCSPs. They model the projection opportunities of table cost functions into an integer linear program. By optimizing the lower bound with linear relaxation, the maximum lower bound that can be inferred is approximated. Immediate future work is to extend our framework to OSAC as a higher level of consistency.

ACKNOWLEDGEMENT

We thank the anonymous referees for their constructive comments. The work described in this paper was substantially supported by grants (CUHK413710 and CUHK413808) from the Research Grants Council of Hong Kong SAR.

REFERENCES

[1] J. Larrosa, "Node and arc consistency in weighted CSP," in *AAAI'02*, 2002, pp. 48–53.
[2] M. Zytnicki, C. Gaspin, and T. Schiex, "A new local consistency for weighted CSP dedicated to long domains," in *SAC'06*, 2007, pp. 394–398.
[3] J. H. M. Lee and K. L. Leung, "Consistency Techniques for Flow-Based Projection-Safe Global Cost Functions in Weighted Constraint Satisfaction," *Journal of Artificial Intelligence Research*, vol. 43, pp. 257–292, 2012.
[4] M. C. Cooper and T. Schiex, "Arc consistency for soft constraints," *Artificial Intelligence*, vol. 154, no. 1-2, pp. 199–227, 2004.
[5] J. Larrosa, "In the quest of the best form of local consistency for weighted CSP," in *IJCAI'03*, 2003, pp. 239–244.
[6] S. de Givry, F. Heras, M. Zytnicki, and J. Larrosa, "Existential arc consistency: Getting closer to full arc consistency in weighted CSPs," in *IJCAI'05*, 2005, pp. 84–89.
[7] J. H. M. Lee, K. L. Leung, and Y. Wu, "Polynomially Decomposable Global Cost Functions in Weighted Constraint Satisfaction," in *AAAI'12*, 2012, pp. 507–513.
[8] J. H. M. Lee and Y. W. Shum, "Modeling Soft Global Constraints as Linear Programs in Weighted Constraint Satisfaction," in *ICTAI'2011*, 2011, pp. 305–312.
[9] Y. W. Shum, "Consistency Techniques for Linear Global Cost Functions in Weighted Constraint Satisfaction," Master's thesis, The Chinese University of Hong Kong, 2012.
[10] T. Schiex, H. Fargier, and G. Verfaillie, "Valued Constraint Satisfaction Problems: Hard and Easy Problems," in *IJCAI'95*, 1995, pp. 631–639.
[11] M. C. Cooper, "High-order consistency in valued constraint satisfaction," *Constraints*, vol. 10, no. 3, pp. 283–305, 2005.
[12] T. Petit, J.-C. Régin, and C. Bessière, "Specific Filtering Algorithm for Over-Constrained Problems," in *CP'2001*, 2001, pp. 451–463.
[13] L. Wolsey, *Integer Programming*. Wiley, 1998.
[14] C. Bessière, E. Hebrard, B. Hnich, Z. Kiziltan, and T. Walsh, "The SLIDE Meta-Constraint," Technical report, 2007.
[15] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, 1982.
[16] J.-C. Régin, "Combination of Among and Cardinality Constraints," in *CPAIOR'2005*, 2005, pp. 288–303.
[17] N. Beldiceanu and E. Contejean, "Introducing global constraints in CHIP," *Mathematical and Computer Modelling*, vol. 20, no. 12, pp. 97–123, 1994.
[18] J.-C. Régin, "Generalized Arc Consistency for Global Cardinality Constraints," in *AAAI'96*, 1996, pp. 209–215.
[19] C. Solnon, V. Cung, A. Nguyen, and C. Artigues, "The Car Sequencing Problem: Overview of State-of-the-Art Methods and Industrial Case-Study of the ROADDEF'2005 Challenge Problem," *European Journal of Operational Research*, vol. 191, no. 3, pp. 912–927, 2008.
[20] A. M. Koster, "Frequency Assignment: Models and Algorithms," Ph.D. dissertation, University of Maastricht, 1999.
[21] M. C. Cooper, S. de Givry, and T. Schiex, "Optimal Soft Arc Consistency," in *IJCAI'07*, 2007, pp. 68–73.