Modeling Soft Global Constraints as Linear Programs in Weighted Constraint Satisfaction

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

***Abstract*—The solving of Weighted CSP (WCSP) with global constraints relies on powerful consistency techniques, but enforcing these consistencies on soft global constraints is not a trivial task. Lee and Leung suggest that a soft global constraint can be used practically if we can find its minimum cost and perform projections/extensions on it in polynomial time, at the same time projections and extensions should not destroy those conditions. However, there are many useful constraints, whose minimum costs cannot be found in polynomial time.**

In this paper, we propose a special class of soft global constraints which can be modeled as integer linear programs. We show that they are soft linear projection-safe and their minimum cost can be computed by integer programming. By linear relaxation we can avoid the exponential time taken to solve the integer programs, as the approximation of their actual minimum costs can be obtained to serve as a good lower bound in enforcing the approximated consistency notions. While less pruning can be done, our approach allows much more efficient consistency enforcement, and we demonstrate the efficiency of such approaches experimentally.

***Keywords*-Constraint Optimization; Global Constraints; Soft Constraint Satisfaction;**

I. Introduction

Weighted Constraint Satisfaction Problems (WCSPs) provide a framework for modeling over-constrained problems and problems with preferences. The basic solution technique for WCSPs is branch-and-bound search augmented with various forms of consistencies [1]–[3]. In addition, a good library of global constraints is essential for us to model complex real-life problems. The key concern with implementing soft global constraints is tractability. Lee and Leung [4] suggest three requirements for a soft global constraints to be practical. First, computation of the minimum cost must be efficient. Second, projections and extensions on the constraints can be performed efficiently. Third, projections and extensions on the constraints will not destroy the last two efficiency requirements. This is called *projection safety* [4]. Lee and Leung further demonstrate that flow-based [5] soft global constraints satisfy the first two requirements, and give instances that are flow-based projection-safe.

Our goal is to introduce more practical soft global constraints into the existing catalog. Many soft global constraints are, however, not flow-based. An example is the soft DISJUNCTIVE constraint, which schedules jobs without overlapping in a non preemptive scheduling problem. Known algorithms for computing the minimum cost of the constraint are exponential.

We propose a special class of soft global constraints which can be modeled as integer linear programs (ILPs), and call them *soft linear constraints*. Thus, minimum costs of such constraints can be computed using integer programming techniques, which are unfortunately exponential in the worst case in general. We propose to compute instead the minimum costs of the linear relaxation of the integer programs, since linear programming algorithms have excellent average case behavior. The relaxed minimum costs can serve as a good lower bound and approximation of the actual minimum costs. We propose constant time algorithms to perform projections and extensions on soft linear constraints, and give sufficient conditions guaranteeing a soft linear constraint to be soft linear projection-safe. By propagating using the approximated minimum costs, we can approximate the various consistency notions for soft global constraints. The tradeoff is less pruning but much more efficient (approximated) consistency enforcement.

We give examples of useful soft linear constraints and show them to be soft linear projection-safe. They include the SOFT_SLIDINGSUM, SOFT_EGCC, and SOFT_DISJUNCTIVE constraints, which are naturally NP-hard to tackle. We perform detailed experimentation to demonstrate the feasibility and efficiency of our proposal.

II. Background

A *weighted constraint satisfaction problem* (WCSP) [6] is a tuple (X, D, C, k). X is a set of $variables$ $\{x_1,x_2,\ldots,x_n\}$. Each variable has its finite $domain$ $D(x_i) \in D$ of values that can be assigned to it. Each variable can only be assigned by one value in its corresponding domain. An assignment on a set of variables can be represented by a tuple ℓ. We denote $\ell[x_i]$ as the value assigned to x_i, $\ell[S]$ as the tuple formed from the assignment on variables in the set S, and $L(S)$ denotes a set of tuples corresponding to all possible assignments on variables S. C is a set of *constraints*. Each $C_S \in C$ over a set of variables $S \subseteq X$ represents a cost

function mapping tuples $\ell[S]$ to a cost valuation structure $V(k) = ([0 \dots k], \oplus, \leq)$. The structure $V(k)$ contains a set of integers $[0, \dots, k]$ with standard integer ordering $\leq$. Addition $\oplus$ is defined by $a \oplus b = \min(k, a + b)$. The subtraction $a \ominus b$ for $a, b \in [0 \dots k]$ and $a \geq b$ is defined as $a \ominus b = a - b$ if $a \neq k$ and $k \ominus a = k$ for any a.

S is the *scope* of C_S, which is the set of variables involved in C_S. We use C_{x_i} to denote a unary constraint over a variable x_i, and $C_\varnothing$ to define the constraint over the empty set of variables, which denotes the least cost of any solution of a WCSP must take. If they are not defined, we assume $C_{x_i}(v) = 0$ for all $v \in D(x_i)$ and $C_\varnothing$ = 0. The cost of a tuple ℓ in a WCSP corresponding to an assignment on X is defined as:

$$cost(\ell) = C_\varnothing \oplus \bigoplus_{C_S \in C} C_S(\ell[S])$$

A tuple ℓ corresponding to an assignment on X 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 a WCSP.

A $global\ constraint$ [7] has a particular semantic and can be represented in a compact way instead of explicitly being enumerated as a table. In addition, for each global constraint there can be more than one cost measure. A separate cost function μ is given to a constraint in case there are more than one such function. For simplicity, we assume that when the cost function is specified, $C_S(\ell)$ is used to represent the constraint C_S associated with the cost function μ, and ℓ is an assignment to S. Also, we use $\min(C_S)$ to denote the minimum cost returned by a constraint C_S.

WCSPs are solved with basic branch-and-bound search augmented with consistency techniques such as NC* [8], (strong) ∅IC [4], [9], (G)AC* [8], FD(G)AC* [2], and (weak) ED(G)AC* [3], [10]. Efficient algorithms to enforce them have also been proposed for unary, binary, and ternary constraints, which involve finding the minimum costs of the constraints, and moving those costs between constraints by *projections* and *extensions* [11]. Projections move costs from n-nary constraints to unary constraints, and extensions are the inverse of projections. A projection of a cost α from C_S to $C_{x_i}(a)$ is a transformation of (C_S, C_{x_i}) to (C'_S, C'_{x_i}) such that if $\ell[x_i] = a$, $C'_S(\ell) = C_S(\ell) \ominus \alpha$, otherwise $C'_S(\ell) = C_S(\ell)$; If $v = a$, $C'_{x_i}(v) = C_{x_i}(v) \oplus \alpha$, otherwise $C'_{x_i}(v) = C_{x_i}(v)$.

To enforce those consistency techniques on high-arity soft global constraints efficiently, Lee and Leung [4] define *projection-safety*. A soft constraint C_S is T *projection-safe* if (a) C_S satisfies property T, and (b) C'_S satisfies property T, where C'_S is obtained from C_S by a valid sequence of projection or extension operations. In other words, the property T is preserved on C_S under projections and extensions. When T is some property satisfied by C_S, e.g. *flow-based*, and C_S is flow-based projection-safe, then C'_S is also flow-based projection-safe, where C'_S is obtained from C_S by a valid sequence of projection or extension operations.

Wu [12] further define *tractable projection-safety*. A constraint C_S is *tractable* if there exists an algorithm to compute $\min(C_S)$ and it runs in time polynomial to its representation size. A soft constraint C_S is *tractable projection-safe* if: (a) C_S is tractable, and (b) C'_S is tractable, where C'_S is obtained from C_S by a valid sequence of projection or extension operations. If a constraint C_S is tractable projection-safe, some consistency techniques like GAC*, FDGAC*, and etc., can always be enforced on C_S in polynomial time.

In this paper, we will formulate soft constraints as *integer linear programs* (ILP) [13]. An ILP P consists of a set of integer variables V, a set of problem constraints Q and an objective function F. Let $V = \{c_1, c_2, \dots, c_n\}$, a problem constraint has the form of $\sum_i^n a_i c_i \leq b$, where $\{a_1, a_2, \dots, a_n\}$ is a set of coefficients and b is the right-hand-side coefficient. An objective function has the form of $\sum_i^n d_i c_i$, where $\{d_1, d_2, \dots, d_n\}$ is a set of coefficients. Solving an ILP is to find values for the variables $\{c_1, c_2, \dots, c_n\}$ minimizing (or maximizing) the objective function F while satisfying all the problem constraints Q.

An *assignment* γ represents the values taken by the variables V. A *feasible solution* is an assignment γ that satisfies all problem constraints Q. An *optimal feasible solution* is an assignment γ representing a feasible solution and the objective function F gives the minimal value. We use obj(P) to denote the value of the objective function from an optimal feasible solution of P.

An integer linear program restricts that every variable in V must take integral value, and solving integer linear programs requires exponential time. By *linear relaxation* [13], every variable in V is allowed to take any real number subject to the problem constraints Q, and the relaxed problem can be solved more efficiently.

III. Soft Linear Constraints

Definition 1: A soft constraint C_S on the set of variables S is a *soft linear constraint* if it can be represented by an integer linear program P, such that $\min(C_S) =$ obj(P).

We use the SOFT_SUM constraint as an example. A SOFT_SUM(S, lb, ub) constraint restricts the sum of the values taken by a set of variables S between a lower bound lb and an upper bound ub.

Definition 2: Given a tuple ℓ, SOFT_SUM(S, lb, ub) holds if $lb \leq \sum_{x_i \in S} \ell[x_i] \leq ub$, where $\ell[x_i]$ is the value assigned to x_i in the tuple ℓ.

We can define the violation measure of SOFT_SUM constraint as that of the SOFT_AMONG constraint given by Maher *et al.* [14]. Given an assignment tuple ℓ on variables S,

$$\text{SOFT\_SUM}(S, lb, ub)(\ell) = \max(\sum_{x_i \in S} \ell[x_i] - ub, lb - \sum_{x_i \in S} \ell[x_i], 0)$$

We give the proof that SOFT_SUM constraint is a soft linear constraint by showing that we can construct a linear program P to express it. For each variable $x_i \in S$, we create a variable c_{x_i} in P which has the same domain as x_i. Two new variables ℓ and u are used to represent the costs arising from violating the lower and upper bounds respectively.

Theorem 1: The SOFT_SUM constraint is a soft linear constraint.

Proof: The SOFT_SUM(S, lb, ub) constraint can be expressed by an integer linear program P where P is defined as:

$$\min \ \ l + u, \ \ s.t.$$

$$lb \leq \sum_{x_i \in S} c_{x_i} - l + u \leq ub$$

$$\min_{v \in D(x_i)} v \leq c_{x_i} \leq \max_{v \in D(x_i)} v, \ \forall x_i \in S$$

$$l \geq 0, u \geq 0$$

such that $\min(\text{SOFT\_SUM}(S, lb, ub)) = \text{obj}(P)$. ∎

Example 1: Consider the constraint C_S = SOFT_SUM$(S, 7, 8)$ where $S = \{x_1, x_2, x_3\}$, $D(x_1) = \{1, 2, 3\}$, $D(x_2) = \{2, 3\}$, and $D(x_3) = \{3\}$, The corresponding integer linear program P is:

$$\min \ \ l + u, \ \ s.t.$$

$$7 \leq c_{x_1} + c_{x_2} + c_{x_3} - l + u \leq 8$$

$$1 \leq c_{x_1} \leq 3, \ \ 2 \leq c_{x_2} \leq 3, \ \ 3 \leq c_{x_3} \leq 3$$

$$l \geq 0, u \geq 0$$

and obj(P) gives the minimum cost of C_S. In this example, obj$(P) = 0$ which equals to $\min(C_S)$.

A. Projections and extensions in Soft Linear Constraints

A T projection-safe constraint preserves its property T after projections and extensions. For example, T can be flow-based and Lee and Leung give examples of flow-based projection-safe constraints [4]. If the minimum cost of a T projection-safe constraint can be computed efficiently, its minimum cost can still be computed efficiently through the consistency enforcement and so it is feasible to use such a constraint in WCSPs.

In this paper, we are interested in *soft-linear* as the property T and we define *soft linear projection-safe constraints*. First we give the sufficient conditions to determine whether a constraint is a soft linear projection-safe constraint, and then we show that given a soft linear projection-safe constraint, the minimum cost can still be computed by solving its corresponding integer linear program after projections and extension.

Lemma 1: Given a constraint C_S which satisfies the following three conditions:

1) C_S is a soft linear constraint and has the corresponding integer linear program P;
2) there exists a function Λ' mapping each optimal feasible solution γ in P to each tuple $\ell \in L(S)$, where $L(S)$ denotes the set of tuples corresponding to all possible assignments on variables S, and;
3) for each value $d \in D(x_i)$ in each variable $x_i \in S$, there exists a 0-1 variable $c_{x_i,d} \in V$ in P such that if $\ell = \Lambda'(\gamma)$ for a feasible optimal solution γ in P and a tuple $\ell \in L(S)$, whenever $\ell[x_i] = d$ for some tuple ℓ, $\gamma[c_{x_i,d}] = 1$; whenever $\ell[x_i] \neq d$, $\gamma[c_{x_i,d}] = 0$

Suppose C'_S is obtained from projecting α from C_S to $C_{x_i}(v)$, or extending α from $C_{x_i}(v)$ to C_S, then C'_S also satisfies these conditions.

Proof: (sketch) We only prove the part for projection as extension is the inverse of projection and so the proof is similar. We first show that C'_S is also a soft linear constraint (condition 1)). Assume P is the corresponding integer linear program of C_S such that there exists an optimal feasible solution γ for P corresponding to an assignment ℓ for C_S. Given the third condition, there exists a variable in $c_{x_i,v}$ in P such that whenever $\ell[x_i] = v$ for some tuple ℓ, $\gamma[c_{x_i,v}] = 1$; whenever $\ell[x_i] \neq v$, $\gamma[c_{x_i,v}] = 0$. After the projection α, an additional term $-\alpha c_{i,v}$ can be added to the objective function F of P. The resulting P' is the corresponding integer linear program of C'_S, since obj(P') = obj$(P) - \alpha c_{i,v} = \min\{C_S\} - \alpha c_{i,v} = \min\{C'_S\}$.

Since P' has the same set of variables $V' = V$ and linear constraints $Q' = Q$ as P has, C'_S also satisfies the conditions 2) and 3). ∎

Lemma 1 implies that if a soft linear constraint satisfies conditions 2) and 3), those conditions are preserved throughout a series of projections and extensions. From Lemma 1, we can give the sufficient conditions of a soft linear projection-safe constraint.

Theorem 2: If a soft global constraint C_S satisfies the conditions stated in Lemma 1, it is a soft linear projection-safe constraint.

Proof: follows directly from Lemma 1. ∎

Theorem 2 gives a sufficient condition for a soft global constraint to be a soft linear projection-safe constraint. In order to construct the corresponding integer linear program P such that the conditions of a soft linear projection-safe constraint can be satisfied, binary variables $c_{x_i,d}$ are introduced for every value d in the domain $d \in D(x_i)$ of every variable $x_i \in S$ in P; for each variable $x_i \in S$, there is an extra linear constraint $\sum_{j \in D(x_i)} c_{x_i,j}$ added to P such that only a value can be assigned to each variable x_i in C_S. According to condition 3), we can easily define Λ'.

We use the SOFT_SUM constraint as an example. Let X = $\{x_1, x_2, x_3\}$ with $D(x_1) = D(x_2) = D(x_3) = \{1, 2\}$, we have the following integer linear program P for C_S = SOFT_SUM$(X, 7, 8)$:

$$\min \ \ l + u, \ \ s.t.$$

$$7 \leq c_{x_1,1} + 2c_{x_1,2} + c_{x_2,1} + 2c_{x_2,2} + c_{x_3,1} + 2c_{x_3,2} - l + u \leq 8$$

$$c_{x_1,1} + c_{x_1,2} = 1, \;\; c_{x_2,1} + c_{x_2,2} = 1, \;\; c_{x_3,1} + c_{x_3,2} = 1$$

$$l \geq 0, u \geq 0$$

where $c_{x_i,d} = \{0, 1\}$ for every $x_i \in S$ and $d \in D(x_i)$.

To find $\min(C_S|x_1 = 1)$, we fix $c_{x_1,1} = 1$ in P. A feasible optimal solution $(c_{x_1,1}, c_{x_1,2}, c_{x_2,1}, c_{x_2,2}, c_{x_3,1}, c_{x_3,2}) = (1, 0, 0, 1, 0, 1)$ is obtained with obj$(P) = 2$, which represents $(x_1, x_2, x_3) = (1, 2, 2)$ with a cost of 2.

If a cost of 2 is projected from C_S to $C_{x_1}(1)$, a new integer linear program P' is constructed with the same set of problem constraints and variables $Q' = Q$, $V' = V$, and F' = $-2c_{x_1,1}$ + F where a new term is added to the objective function, such that $F' = l + u - 2c_{x_1,1}$.

By assigning $(x_1, x_2, x_3) = (1, 2, 2)$ back to P', obj(P') becomes 0 which is the cost of this tuple after projection. We can see that projecting and extending cost to soft linear projection-safe constraint C_S requires constant time.

We cannot convert all the soft linear constraints into soft linear projection-safe constraints directly. We use the SOFT_NVALUE constraint as an example. The SOFT_NVALUE(S, lb, ub) constraint restricts the number of distinct values taken by a set of variables S between a lower bound lb and an upper bound ub. We define $D(S) = \bigcup_{x_i \in S} D(x_i)$. Given an assignment tuple ℓ on variables S,

$$\text{SOFT\_NVALUE}(S, lb, ub)(\ell) = \max(k - ub, lb - k, 0)$$

where

$$k = \sum_{d \in D(S)} \min(|\{i|\ell[x_i] = d \; \forall x_i \in S\}|, 1)$$

An integer linear program P can be constructed such that $\min(\text{SOFT\_NVALUE}(S, lb, ub)) = \text{obj}(P)$:

$$\min \;\; l + u, \;\; s.t.$$

$$\sum_{d \in D(x_i)} c_d \geq 1, \quad \forall x_i \in S$$

$$lb \leq \sum_{d \in D(S)} c_d - l + u \leq ub; \; c_d = \{0, 1\} \quad \forall d \in D(S)$$

and SOFT_NVALUE is a soft linear constraint. Since all the variables $x_i \in S$ share the same set of 0-1 variables $c_d \in V$ of P, more than one tuple in C_S can be represented by an optimal feasible solution γ of P. So the SOFT_NVALUE constraint with our formulation does not satisfy the condition 2) of Lemma 1. By naively adding variables to satisfy that condition, it becomes a quadratic program which does not satisfy the condition 1) of Lemma 1. We have yet to find the linear formulation such that we can model SOFT_NVALUE as a soft linear projection-safe constraint.

B. Linear Relaxation of Integer Linear Program

A requirement for a constraint C_S to be a tractable projection-safe soft constraint is that its minimum cost $\min(C_S)$ can be computed efficiently. However, given a soft linear projection-safe constraint C_S, we compute its minimum cost $\min(C_S)$ by solving the corresponding integer linear program P, which is an NP-hard problem. As $\min(C_S)$ is used in enforcing different consistency techniques, we propose that by solving P with linear relaxation, we can obtain a good lower bound for $\min(C_S)$ such that an approximation of the consistency techniques can be enforced. In this way we relax an NP-hard optimization problem into a related problem which is solvable in polynomial time.

Here we assume that for every integer linear program P used in soft linear projection-safe constraints in WCSP is a minimization problem.

Theorem 3: Suppose P is an integer linear program corresponding to a soft linear projection-safe constraint C_S and P is a minimization problem, and relaxed_obj(P) is the value of the objective function from a feasible optimal solution of P under linear relaxation, then: a) relaxed_obj$(P) \leq$ obj(P), and b) Cost can be projected equals to $\max(\lceil \text{relaxed\_obj}(P) \rceil, 0)$.

Proof: Given a feasible solution γ' of P corresponding to ℓ', γ' must be a feasible solution of P under linear relaxation. So obj(P) cannot be smaller than relaxed_obj(P). As it is possible to take real values in the relaxed problem, we can have relaxed_obj(P) smaller than obj(P).

To project α from C_S to $C_{x_i}(v)$ on a soft linear projection-safe constraint, there must exist an integer linear program P where obj$(P(c_{x_i,v} = 1)) = \alpha$. Since obj$(P(c_{x_i,v} = 1))$ is an integral value, solving P by linear relaxation obtains a minimum cost relaxed_obj$(P(c_{x_i,v} = 1))$ to be projected; given $\lceil$relaxed_obj$(P(c_{x_i,v} = 1))\rceil \leq$ obj$(P(c_{x_i,v} = 1))$, it must be feasible to project $\lceil$relaxed_obj$(P(c_{x_i,v} = 1))\rceil$ to $C_{x_i}(v)$. At the same time as obj$(P(c_{x_i,v} = 1)) \geq 0$ given that all the projections and extensions performed previously are feasible. Even relaxed_obj$(P) < 0$, we can still take the cost can be projected as 0. ■

We use GAC* as an example. GAC* is a consistency notion used in WCSPs, which requires that in each value of each variable, there must exists a supporting tuple with its cost $= 0$ in each constraint related to that variable.

Definition 3: Given a WCSP $P(X, D, C, k)$, a value $x_i(a)$ where $a \in D(x_i)$ and $x_i \in X$ is GAC* with respect to constraint $C_S \in C$ if it is NC* and there is a tuple $\ell \in L(S)$ with $\ell[x_i] = a$ such that $C_S(\ell) = 0$. A value $x_i(a)$ is NC* if $C_\varnothing \oplus C_{x_i}(a) < k$.

So if $\min(C_S(x_i = a)) = 0$, there exists a supporting tuple which is the tuple gives that cost.

By linear relaxation, we have an approximation of $\min(C_S)$ from relaxed_obj(P), where P is the correspond-

ing linear program of C_S. We can define an approximation of GAC* by relaxing the requirements of GAC*.

Definition 4: $x_i(a)$ is approximated GAC* with respect to a soft linear projection-safe constraint $C_S \in C$ if it is NC* and relaxed_obj$(P|c_{x_i,a} = 1) \leq 0$, where P is the corresponding linear program of C_S.

This approach also allows us to define the approximation of some stronger consistency techniques like FDGAC* and weak EDGAC*.

IV. Modeling Soft-Global Constraints as Linear Constraints

In this section we introduce three soft global constraints, include the Soft_SlidingSum, Soft_eGCC, and Soft_Disjunctive/Cumulative constraints. We show that computing their minimum cost is NP-hard by showing that enforcing GAC [15], a consistency notion in classical CSPs, on the related hard constraint is NP-hard. Since GAC* collapses to GAC when WCSPs collapse to CSPs [16]. We can enforce GAC on the related hard constraint of C_S in polynomial time if there exists a polynomial time algorithm to find the minimum cost of C_S. On the other hand, if the enforcement of GAC on a related hard constraint of C_S is NP-hard, finding the minimum cost of C_S must be NP-hard. By linear relaxation, we can obtain the approximated lower bounds and use them in enforcing the approximation of the consistency techniques.

A. Soft_SlidingSum *Constraint*

The SlidingSum$(S, [p_1, \ldots, p_m])$ constraint [14] takes a sequence of n variables $S = \{x_1, \ldots, x_n\}$ and m windows. For every window $p_i = \{lb_i, ub_i, k_i, s_i\}$, the sum of the variables is restricted between a lower bound lb_i and an upper bound ub_i from its starting position s_i for a length k_i. Enforcing GAC on Sum constraint is NP-hard [17]. As the SlidingSum constraint can can be represented by a conjunction of multiple Sum constraints, enforcing GAC on SlidingSum is NP-hard.

Definition 5: The SlidingSum$(S, [p_1, \ldots, p_m])$ constraint holds iff $lb_i \leq \sum_{j=s_i}^{s_i+k_i-1} x_j \leq ub_i$ for every i from 1 to m.

We can define the violation measure of the Soft_SlidingSum constraint as that of the Soft_Among constraint [14]. Given an assignment tuple ℓ on variables S,

$$\text{Soft\_SlidingSum}(S, [p_1, \ldots, p_m])(\ell)$$
$$= \sum_{j=1}^{m} \max(\sum_{h=s_j}^{s_j+k_j-1} \ell[x_h] - ub_j, lb_j - \sum_{h=s_j}^{s_j+k_j-1} \ell[x_h], 0)$$

Computing the minimum cost of Soft_SlidingSum according to the violation measure defined here is NP-hard as we can reduce a Soft_SlidingSum constraint to a SlidingSum constraint. We can model this constraint as a soft linear projection-safe constraint such that we can compute the approximated minimum cost efficiently by linear relaxation.

Theorem 4: The Soft_SlidingSum constraint is a soft linear projection-safe constraint.

Proof: The Soft_SlidingSum$(S, [p_1, \ldots, p_m])$ constraint can be expressed by an integer linear program P where P is defined as:

$$\min \sum_{j=1}^{m} l_j + u_j, \;\; s.t.$$
$$lb_j \leq \sum_{h=s_j}^{s_j+k_j-1} \sum_{d \in D(h)} d * c_{x_h,d} - l_j + u_j \leq ub_j, \quad \forall j = 1 \ldots m$$
$$l_j \geq 0, u_j \geq 0, \quad \forall j = 1 \ldots m$$
$$\sum_{d \in D(x_i)} c_{x_i,d} = 1, \; \forall i = 1 \ldots n; \; c_{x_i,d} = \{0, 1\}, \; \forall x_i \in S, d \in D(x_i)$$

If $x_i = d$, $c_{x_i,d} = 1$; otherwise $c_{x_i,d} = 0$. By Theorem 2, Soft_SlidingSum constraint is a soft linear projection-safe constraint. ■

B. Soft_eGCC *constraint*

The eGCC(S_X, S_Y) constraint [18] is defined for two sets of $n + m$ variables S_X and S_Y where $S_X = \{x_1, \ldots, x_n\}$ is a set of assignment variables and $S_Y = \{y_{d_1}, \ldots, y_{d_m}\}$ is a set of counting variables. The idea is that each value d_j where $y_{d_j} \in S_Y$ is used exactly y_{d_j} times by the variables S_X. Enforcing GAC on every variable of eGCC is NP-hard [18].

Definition 6: The eGCC(S_X, S_Y) constraint holds iff $y_{d_i} \in D(d_i) \wedge occ(d_i, (x_1, \ldots, x_n)) = y_{d_i}$ for every d_i where $y_{d_j} \in S_Y$.

where $occ(v, t)$ is the number of occurrences of v in t.

We can define the Soft_eGCC constraint with the same violation measure as variable-based violation measure used in Soft_GCC [5]. The constraint is softened by allowing counting variables $y_{d_i} \in S_Y$ to take values other than $occ(d_i(x_1, \ldots, x_n))$. Given an assignment tuple ℓ in variables $S = S_X \cap S_Y$,

$$\text{Soft\_eGCC}(S_X, S_Y)(\ell)$$
$$= \sum_{j=1}^{m} |y_{d_j} - occ(d_j, (x_1, \ldots, x_n))|$$

Computing the minimum cost of Soft_eGCC according to the violation measure defined here is NP-hard as we can reduce a Soft_eGCC constraint to a eGCC constraint. We can model this constraint in the form of a soft linear projection-safe constraint such that we can compute the approximated minimum cost efficiently by linear relaxation.

Theorem 5: The Soft_eGCC constraint is a soft linear projection-safe constraint.

The proof is similar to that of theorem 5.

C. SOFT_DISJUNCTIVE/CUMULATIVE *Constraint*

The DISJUNCTIVE$(S, p_1, \ldots, p_n)$ constraint [19] is used in non-preemptive scheduling. A set of n variables $S = x_1, \ldots, x_n$ is used to represent the beginning time of n jobs. Each job $x_i \in S$ has its process time p_i and its possible start time defined by its domain $d \in D(x_i)$. After one job has started, it cannot be interrupted to start processing another job. DISJUNCTIVE constraint restricts any two jobs from processing at the same time. Enforcing GAC on DISJUNCTIVE is NP-hard [20].

Definition 7: The DISJUNCTIVE$(S, p_1, \ldots, p_n)$ constraint holds if $(x_i + p_i \leq x_j) \vee (x_j + p_j \leq x_i)$ for every pair of $x_i, x_j \in S$ [19]

The penalties for earliness and tardiness can be modeled by unary costs of the values representing the start time. Here we only define the violation measure to include the penalties of jobs being processed at the same time.

This constraint is softened by allowing more than one job to be processed at the same time with a cost as the penalty. Such a violation measure allows SOFT_CUMULATIVE to be modeled with the same formulation, where the CUMULATIVE constraint allows k jobs to be processed at the same time instead of 1 in the DISJUNCTIVE constraint.

First we define T which consists of every possible start time $d \in D(x_i) \forall x_i \in S$ and all possible finish time $p_i + d \in D(x_i) \forall x_i \in S$. Given an assignment tuple ℓ in variables S,

$$\text{SOFT\_DISJUNCTIVE}(x_i, \ldots, x_n, p_1, \ldots, p_n)(\ell)$$
$$= \sum_{t=0}^{T} \sum_{i=1}^{n} \max(|\{i | \ell[x_i] \leq t \leq \ell[x_i] + p_i\}| - 1, 0)$$

Computing the minimum cost of SOFT_DISJUNCTIVE according to the violation measure defined here is NP-hard as we can reduce a SOFT_DISJUNCTIVE constraint to a DISJUNCTIVE constraint. We can model this constraint in the form of a soft linear projection-safe constraint such that we can compute the approximated minimum cost efficiently by linear relaxation.

Theorem 6: The SOFT_DISJUNCTIVE constraint is a soft linear projection-safe constraint.

The proof is similar to that of theorem 5.

V. EXPERIMENTAL RESULTS

To demonstrate the practicality of our framework, we implement the constraints described in section IV in Toulbar2 v0.9, and use IBM ILOG CPLEX Optimizer 12.2 to solve the related linear problems. We compare the results of models with linear constraint and different modeling methods with different levels of consistency including strong ∅IC [4], (G)AC* [8], and FD(G)AC* [2]. Approximated version of those consistency levels are used for soft linear projection-safe constraints.

In this experiment, variables with smaller domains and values with lower unary costs are assigned first. The tests 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. In each benchmark we use different timeout and report the number of solved instances #s, 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 program solver ILOG. We truncate the floating point variables in ILOG at the 10-th decimal place. We first compare the number of solved cases. The best result among those with the most cases solved is highlighted in bold.

A. The generalized car sequencing problem (generalizing prob001 in CSPLib)

We evaluate performance of the SOFT_SLIDINGSUM constraint by comparing the performance of our linear constraint approach with the modeling by flow-based soft constraints. Given n cars of $u \in U$ different types, and a set of options I which each type may or may not be equipped with, each assembly line of an option $i \in I$ allows a maximum of m_i cars for every s_i cars with that option equipped to be built. A car-sequencing problem is to find a sequence for the cars to be built such that all the constraints above are satisfied. We generalize the problem such that a random cost $c_{u,i}$ is required for each type of car $u \in U$ to equip each option $i \in I$, and each assembly line of an option $i \in I$ allows a maximum of m_i costs to be spent on that option for every s_i cars in total.

We use n variables with domain from 1 to u to represent the type of the n-th car to be built in the sequence. A SOFT_EGCC constraint is used to ensure that the number of cars of each type is built according to the plan. A SOFT_SLIDINGSUM constraint is used to ensure that at most m_i resources are used out of s_i cars for every option $i \in I$. We soften the problem as SOFT_SLIDINGSUM allows violation. We generate 10 problems for each parameters. For each set of parameter, there are n cars, 5 options, and u types, and there are $1/2$ chance for each option to be equipped by each type, and a random amount of resource requirement is generated for each option of each type.

We compare the performance of our implementation against the model with SOFT_REGULAR constraints, which are flow-based projection-safe soft constraints and they can be used to model the SOFT_SUM constraints and form the SOFT_SLIDINGSUM constraints.

Results are shown in Table I. In this benchmark, models using linear constraints run faster and prune more than models with SOFT_REGULAR constraints in most cases. FDGAC* is also faster than than GAC* and strong ∅IC in models with a linear constraint.

B. The magic series problem (prob019 in CSPLib)

We evaluate performance of the SOFT_EGCC constraint on magic series problems. A non-empty finite series $S = (s_0, s_1, \ldots, s_n)$ is *m*agic if and only if there are s_i occurrences of $i \in S$ for each integer i ranging from 0 to n. For example, $S = (3, 2, 1, 1, 0, 0, 0)$ is an example of a magic series for $n = 6$ as there are three 0's, two 1's, a 2, a 3, and no 4, 5, and 6 in the series S. The hard version of this problem with the size of n can be modeled by n EGCC constraints. We soften the problem by using the SOFT_EGCC constraints which allow that the occurrences of i are not equal to the value of s_i, and unary costs are assigned randomly to each value of s_i for all variables $s_i \in S$.

We compare our model with the model constructed by decomposing the SOFT_EGCC constraints into SOFT_AMONG constraints with counting variables modeled by SOFT_GCC constraints. GAC* can be enforced in polynomial time in each SOFT_AMONG constraint [16].

Results are shown in Table II. In this benchmark, models using linear constraints always prune more than models with SOFT_AMONG constraints. Those models also run faster when stronger consistency techniques like GAC* and FDGAC* are used, but FDGAC* does not benefit enough over GAC* so although the number of backtrack is fewer, the run-time of FDGAC* is worse than GAC* by about 2 times.

C. The weighted tardiness scheduling problem (in OR-Library)

We evaluate performance of the SOFT_DISJUNCTIVE constraint the weighted tardiness scheduling problem. We generate random scheduling problems with different number of jobs n, the average duration of each job d, and the total available time slots t. A SOFT_DISJUNCTIVE constraint is used to ensure no two jobs are processed at the same time. A time slot with the length of $t/2$ is given to each job, and a random earliness/tardiness penalty is given to each job if it cannot be processed within the given time slot. We soften the problem by allowing the jobs to be processed at the same time with a penalty. We compare the result of the linear constraint approach of SOFT_DISJUNCTIVE with the integer programming approach of the same implementation, which allows the exact minimum costs to be found and so the consistency algorithms like GAC*, etc., to be enforced.

Results are shown in Table III. In this benchmark, the integer linear program approach prunes more than linear relaxation approach as the exact minimum costs of each constraint are found in propagation steps. However it also takes much more time to solve and the extra pruning power offered in using integer linear programs does not pay off.

Table I
THE GENERALIZED CAR SEQUENCING PROBLEM USING SOFT_SLIDINGSUM

Modeling with soft linear constraints									
n	strong ∅IC			GAC*			FDGAC*		
	s	bt	time	s	bt	time	s	bt	time
10	10	101	1.18	10	41	0.55	10	**21**	**0.52**
11	10	106	1.61	10	56	0.97	10	**24**	**0.81**
12	10	138	2.51	10	119	1.44	10	**48**	**1.15**
13	10	278	6.88	10	223	4.29	10	**107**	**4.04**
14	10	1743	30.16	10	585	17.63	10	**264**	**13.12**
15	9	1441	17.24	9	458	15.20	10	**503**	**33.16**
16	9	2347	35.77	9	782	25.76	9	**434**	**18.77**
17	9	2802	43.24	9	1029	31.80	9	**674**	**23.85**
18	9	3959	81.95	9	1509	51.43	9	**1390**	**34.52**
Modeling with flow-based soft constraints (SOFT_REGULAR)									
n	strong ∅IC			GAC*			FDGAC*		
	s	bt	time	s	bt	time	s	bt	time
10	10	10966	2.86	10	4023	1.57	10	865	0.72
11	10	34146	9.70	10	14775	5.90	10	4501	3.86
12	10	149251	46.10	10	55866	24.73	10	24497	22.15
13	10	514602	171.45	10	212406	97.06	10	53913	54.57
14	7	378079	148.01	9	279748	152.24	10	104588	108.94
15	4	191455	80.04	4	77584	45.11	10	110117	152.39
16	0	*	*	0	*	*	2	138938	227.10
17	0	*	*	0	*	*	2	188422	309.32
18	0	*	*	0	*	*	0	*	*

Table II
THE MAGIC SEQUENCE USING SOFT_EGCC CONSTRAINTS

Modeling with soft linear constraints									
n	strong ∅IC			GAC*			FDGAC*		
	s	bt	time	s	bt	time	s	bt	time
6	10	73	0.31	10	14	0.09	10	**13**	0.10
9	10	367	2.54	10	23	**0.24**	10	**19**	0.27
12	10	5010	67.44	10	54	**0.71**	10	**44**	0.99
15	3	12136	274.88	10	89	**1.70**	10	**53**	2.32
18	2	29409	508.86	10	93	**3.03**	10	**64**	4.80
Modeling with flow-based soft constraints (SOFT_AMONG)									
n	strong ∅IC			GAC*			FDGAC*		
	s	bt	time	s	bt	time	s	bt	time
6	10	229	0.23	10	48	0.06	10	25	**0.05**
9	10	8878	16.13	10	680	5.00	10	83.4	1.26
12	4	295761	139.88	4	6142	220.22	10	252	19.15
15	0	*	*	0	*	*	10	810	228.03
18	0	*	*	0	*	*	0	*	*

VI. CONCLUSION

We define the class of soft global constraints called *soft linear constraints* which can be modeled as integer linear programs. Our work gives the sufficient conditions guaranteeing a constraint to be soft linear projection-safe and shows the projections and extensions can be done in constant time. We further propose to relax the integer program by linear relaxation. With the excellent average case behavior of linear programming algorithms, we show that while we have the tradeoff of less pruning brought by the approximation, we have a much more efficient consistency enforcement. Our proposal proves practical ways to enforce consistencies on soft versions of SLIDINGSUM(), EGCC(), and DISJUNCTIVE() constraints, which are naturally NP-

Table III
THE WEIGHTED TARDINESS SCHEDULING PROBLEM USING SOFT_DISJUNCTIVE

With linear relaxation									
n,d,t	strong ∅IC			GAC*			FDGAC*		
	s	bt	time	s	bt	time	s	bt	time
3,3,12	10	18	0.08	10	7	**0.05**	10	**6**	0.06
4,4,20	10	56	0.36	10	13	**0.14**	10	**8**	0.18
5,5,30	10	134	1.62	10	35	**0.60**	10	19	0.68
6,5,35	10	797	14.90	10	382	7.01	10	32	**1.90**
7,5,40	10	3303	101.86	10	2253	61.89	10	27	**2.78**
8,5,45	4	5768	221.06	4	3894	135.51	10	**214**	**22.09**
Without linear relaxation									
n,d,t	strong ∅IC			GAC*			FDGAC*		
	s	bt	time	s	bt	time	s	bt	time
3,3,12	10	18	1.79	10	7	1.02	10	**6**	1.37
4,4,20	10	56	13.35	10	13	6.62	10	**8**	7.56
5,5,30	10	134	74.45	10	35	38.28	10	**15**	40.77
6,5,35	10	238	230.80	8	61	100.99	10	**19**	121.82
7,5,40	3	250	429.97	4	81	219.85	10	**23**	302.98
8,5,45	0	*	*	0	*	*	0	*	*

hard to tackle. Through experiments, we show that our approach is competitive in term of run-time and reduction in search space. A. M. Koster [21] propose a generic way of formulating WCSPs into linear programs. While their framework can model every constraint used in WCSPs, our work focuses on the global constraints which has particular semantics. By identifying the specific structure of each constraint, we can model them more efficiently. An immediate future work is to find a generic way to identify whether a constraint is a soft linear (projection-safe) constraint. We will also investigate whether our approach also performs efficiently on other soft linear projection-safe constraints.

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] M. C. Cooper and T. Schiex, "Arc consistency for soft constraints," *Artificial Intelligence*, vol. 154, pp. 199–227, 2004.

[2] J. Larrosa, "In the quest of the best form of local consistency for weighted CSP," in *IJCAI'03*, 2003, pp. 239–244.

[3] 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.

[4] J. H. M. Lee and K. L. Leung, "Towards efficient consistency enforcement for global constraints in weighted constraint satisfaction," in *IJCAI'09*, 2009, pp. 559–565.

[5] W. van Hoeve, G. Pesant, and L. Rousseau, "On global warming: flow-based soft global constraints," *J. Heuristics*, vol. 12, no. 4-5, pp. 347–373, 2006.

[6] T. Schiex, H. Fargier, and G. Verfaillie, "Valued constraint satisfaction problems: hard and easy problems," in *IJCAI'95*, C. Mellish, Ed., Montreal, 1995.

[7] N. Beldiceanu, M. Carlsson, and J. Rampon, "Global constraint catalog," SICS Research Report, 2005.

[8] J. Larrosa, "Node and arc consistency in weighted CSP," in *AAAI'02*, 2002, pp. 48–53.

[9] 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.

[10] J. H. M. Lee and K. L. Leung, "A stronger consistency for soft global constraints in weighted constraint satisfaction," in *AAAI'10*, 2010, pp. 121–127.

[11] M. C. Cooper, "High-order consistency in valued constraint satisfaction," *Constraints*, vol. 10, pp. 283–305, 2005.

[12] Y. Wu, "Tractable projection-safe soft global constraints in weighted constraint satisfaction," Master's thesis, The Chinese University of Hong Kong, 2011.

[13] L. Wolsey, *Integer Programming*. Wiley, 1998.

[14] M. Maher, N. Narodytska, C.-G. Quimper, and T. Walsh, "Flow-based propagators for the sequence and related global constraints," in *CP'2008*, 2008, pp. 159–174.

[15] C. Bessière and J.-C. Régin, "Arc consistency for general constraint networks: preliminary results," in *IJCAI'97*, 1997, pp. 398–404.

[16] K. Leung, "Soft global constraints in weighted constraint satisfaction," Master's thesis, The Chinese University of Hong Kong, 2009.

[17] C. Bessière and P. V. Hentenryck, "To be or not to be ... a global constraint," in *CP'2003*, 2003, pp. 789–794.

[18] I. Katriel and S. Thiel, "Complete bound consistency for the global cardinality constraint," *Constraints*, vol. 10, pp. 115–135, 2005.

[19] J. N. Hooker, *Integrated Methods for Optimization*. Springer Science + Business Media, 2007.

[20] A. Aggoun and N. Beldiceanu, "Extending chip in order to solve complex scheduling and placement problems," *Mathematical and Computer Modelling*, vol. 17, no. 7, pp. 57–73, 1993.

[21] A. M. Koster, "Frequency assignment: models and algorithms," Ph.D. dissertation, University of Maastricht, 1999.